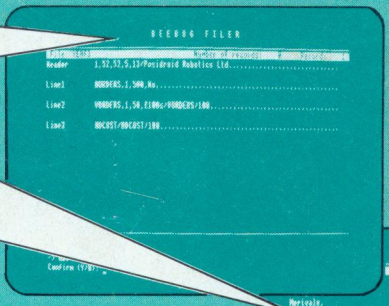
VOLUME 5 NUMBER 2 JUNE 1986 PRICE £1.30

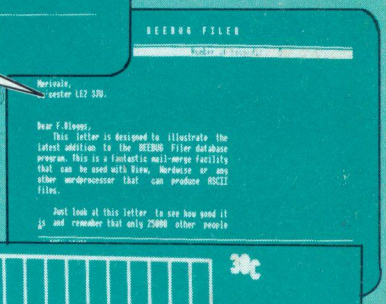
BEEBUG

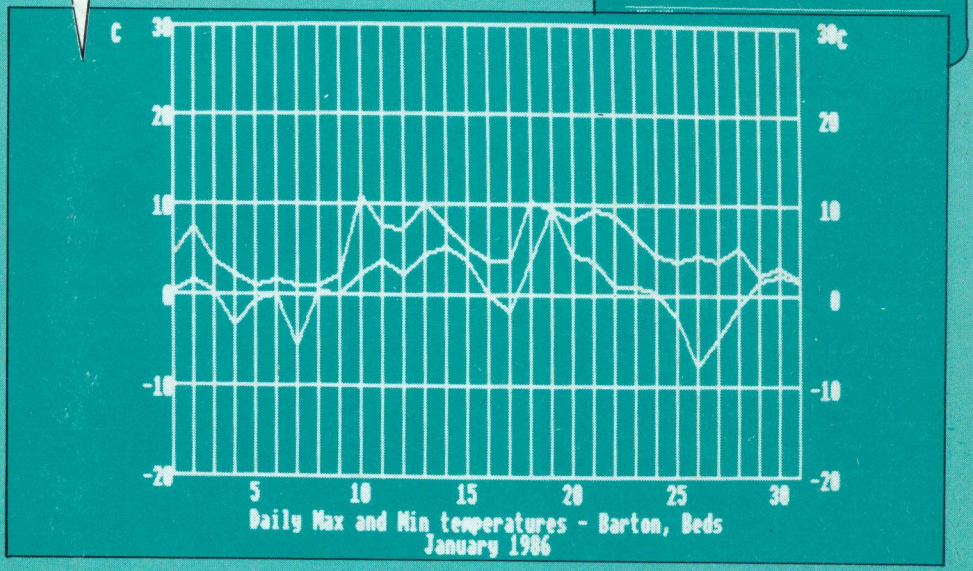
FOR THE BBC MICRO

BEEBUG FILER

Records
Selecting
Sorting
Mail-Merge



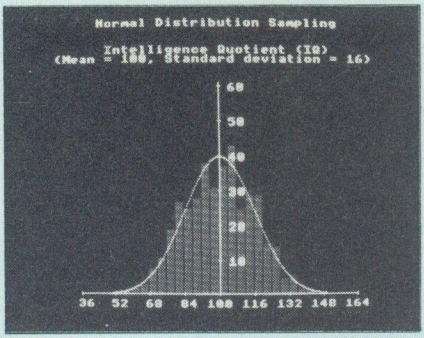




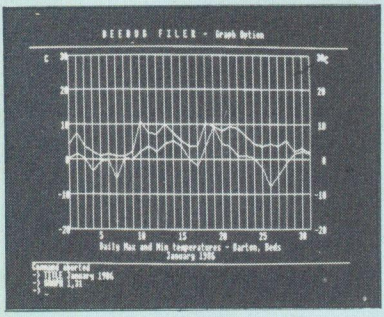
Beebug Filer – Graph Option

BEEBUG

VOLUME 5 NUMBER 2
JUNE 1986

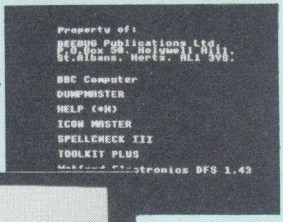


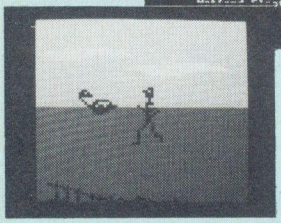
Computer Simulation



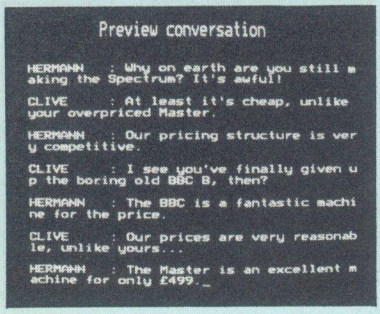
Filer Graphics

Sideways RAM





Movie Maker



Talking Heads

GENERAL CONTENTS

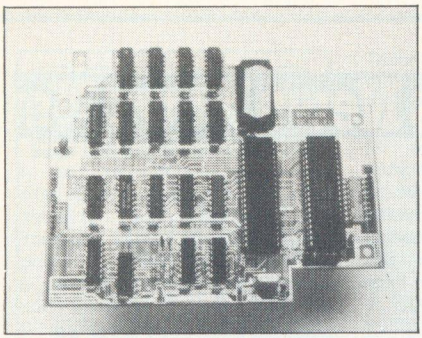
3 Editorial Jottings
4 News
5 BEEBUGSOFT Forum
6 Computer Simulation (Part 1)
9 Supercharging the Master
12 BEEBUG Filer Goes Graphic
16 Wigmore Mouse
19 Debug Wordwise Plus Programs
21 Points arising
22 The Master Series
Opening up the Private RAM
24 Passing Arrays to Procedures
26 Sounds Unnatural
27 Movie Maker
28 BEEBUG Workshop
Polishing Your Programs
30 Software for Sideways RAM (Part 1)
35 Hardware Animation
36 First Course
Using the ADVAL Function
39 Macro Assemblers
42 Postbag
43 Hints and Tips
44 Light Fantastic
45 Talking Heads

PROGRAMS

6 Computer Simulation
11 Filer Graph Program
19 Wordwise Plus Debug Utility
22 Master Series Routines
24 Passing Arrays Examples
28 Workshop Examples
30 Sideways RAM Examples
35 Animation Technique & Demo
36 Examples on Use of ADVAL Function
45 Talking Heads

HINTS AND TIPS

Basic I Relocation
Masking
Pretty Easy List
Half Procedures
Wordwise Plus Sentence Jumper
Decoding USR
High Precision Analogue
Tube or Not

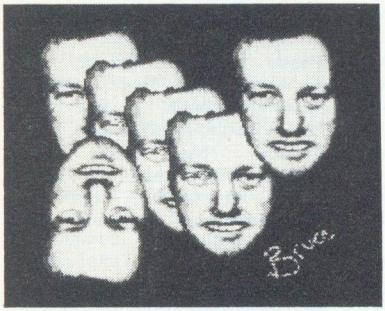


Master Turbo Upgrade

Wordwise Plus Utility

```
WORDWISE DEBUG

C  COMPILE SINGLE STEPPER
D  DISPLAY SOURCE PROGRAM
E  EXIT TO SOURCE PROGRAM
```



Wigmore Mouse

Macro Assemblers

```
; This macro sets up the header for
; a sideways ROM
; requires  1/ language entry
;           2/ service entry
;           3/ ROM type
;           4/ binary version no.
;           5/ ROM title string
;           6/ version string
;           7/ copyright message

SWR         MACRO
offset      DEFL Copy-origin
origin      JMP &1
            JMP &2
            DEFB &3
            DEFB offset
            DEFB &4
            DEFM &5
            DEFB &6
Copy        DEFM "(C)"
            DEFB &7
            DEFB 0
            ■
```

EDITORIAL JOTTINGS

BEEBUG

We are concerned that BEEBUG should continue to reflect the needs and interests of you, the members. Through the magazine, through our range of software products, and more recently through our BEEBUG retail shop, we have tried to provide the services that we believe BEEBUG members want. At all times we do aim to provide a personal service and to provide continuing support, not only for our own products but for all matters related to the BBC micro and the new Master series.

In order that we can continue to improve and expand the range of services that we offer members, we need to increase the membership of BEEBUG. In this task, we would like to enlist your active help and support. If you can persuade just one person to join BEEBUG this year and other members do likewise, then this would help enormously and provide the impetus that we need to expand our range of activities for the benefit of members.

Of course, if you would like to send us your views on how we can improve BEEBUG, then we would be delighted to hear from you. BEEBUG can only remain successful if we can provide what you want from your user group, and we can only do that if you tell us what that is.

MASTER SERIES

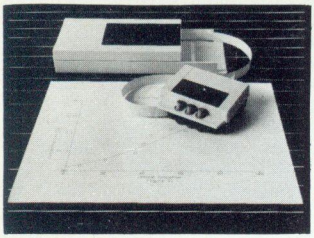
Judging just from the number of enquiries received through our shop, the new Master Series is proving very popular with BEEBUG members. Starting this month our own new Master Series will provide articles and programs just for Master users. In this series we shall be looking at the many new features of the Master and prising open some of the less well documented secrets of Acorn's new range. In addition, this month also sees our in-depth review of the Turbo upgrade for the Master 128, producing one of the fastest micros available from any source

PROGRAM CLASSICATION

All programs in the magazine, and on the magazine cassette/disc, are marked with the symbols shown below. An uncrossed symbol indicates full working, a single line through a symbol shows partial working for that configuration (normally some modifications will be required), and a cross through a symbol indicates a program that will not work on that type of system. There is a symbol for the B+ which includes the 128K version, and a symbol for the new Master series.

Basic I		Electron	
Basic II		Disc	
Tube		Cassette	
Model B+		Master 128	

News News News News News News Ne



Penman Power

The 'Data Plotting and Analysis' package from Penman Products is an inexpensive piece of software to make full use of the Penman Plotter (see the review in BEEBUG Vol.4 No.2). The package includes a data editor, a 1728 cell spreadsheet, and is capable of a wide range of statistical analysis of data such as cross-correlation, confidence limits, sample variance, standard deviation, and so on.

The results are displayed graphically and after previewing them on the screen they are automatically plotted in full colour by the Penman Plotter. The software package costs £45 or £319 for a complete package including the plotter. Details from Penman on 0903-209081.

Parley Vous Peartree?

An ingenious idea from Peartree computers will allow you to write fluent French, German, or Spanish. 'Tick-tack' is a letter writing software package that allows you to write on a Beeb using a range of standard English clauses which it then translates into the language of your choice. The only problem is the price - a rather hefty £160 per language. However, pour les riches, les details sont available from Peartree sur 0480-50595.

The Direkt Approach

If you want a more active approach to communicating with our EEC partners then BBC Soft's latest software releases will help. 'Deutsch Direkt' and 'A Vous La France' are designed to accompany the TV programmes of the same name and make us all fluent in German and French. The packages include as well audio cassettes and booklets and cost £22.95 and £19.95, respectively, from BBC Soft on 01-927 4518.

Arcade Incentives

Incentive Software is launching its latest Beeb game (a licensed version of the arcade game 'Moon Cresta') with an unusual competition - to win an arcade machine featuring the original game. To gain this truck-load of electronics you will have to first spend £7.95 (cassette) or £9.95 (disc) on Moon Cresta, score over 30,000 points, and then come first in the lucky draw. Details from Incentive on 0734-591678.

Two into One

The ingenious Mr. Terrell (see last month's review of sideways RAM modules) has solved the problem of all you BEEBUG members with two printers and one computer or a single printer and a couple of Beebs. His printer switch boxes will connect up such trios allowing switching between routes. A 2 Beeb to 1 printer box will cost you £35 and a 2 printer to 1 Beeb box £38. Details from C.F.Terrell on 04024-71426.

More from Micronet

The main problem with Prestel and other Viewdata systems is that the resolution, and therefore realism, of graphics is decidedly poor. Micronet has skirted around this problem with the introduction of high resolution, 'photo-quality' pictures to accompany news and features related to the BBC micro. These pictures can only be received via your modem from Micronet if you have the right software to decode the data but surprisingly this is free to Micronet members. Details from Micronet on 01-278 3143.

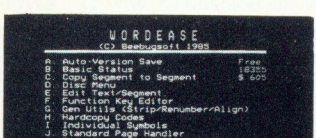
Also from Micronet is an idea that will appeal to many businessmen beleaguered with paperwork. Accountancy company MAS is offering a service in the Biznet area of Micronet whereby all payments and receipts are entered on screen and downloaded to MAS where they are analysed and a hard copy comprising audit and VAT report, purchase and sales ledgers, and so on, is returned to you through the post. Further details from MAS on 0937-63778.

BEEBUG SOFT FORUM

Expanding Wordease

Wordease is written in the Wordwise Plus programming language, and it is a simple matter to expand it to encompass your own routines. As an example, we will show how to incorporate the Wordwise Debug program printed elsewhere in this issue.

The first thing to do is to make yourself a working copy of the Wordease disc, since you may not write to the dual format disc as sold by Beebugsoft. You do this using the *COPY command (rather than *BACKUP), copying all files except for the two whose filenames consist of asterisks.



The modifications required are simple. There are only two minute adjustments required to the Wordease main menu program, and three variables in the Debug need to be globally changed to avoid any clashes.

To do this proceed as follows. Load Wordease as usual with Shift-Break, then press Escape once the menu appears, and alter the following lines:

```
P."M. Mail Merge" to
P."M. Mail Merge      N.
Debug Utility"

IFX$>="A"A.X$<"M"T.G.e to
IFX$>="A"A.X$<"N"T.G.e
```

The two lines are around the sixtieth and ninetieth respectively. When you have done this, save the segment as W.UTILS to your newly copied Wordease disc.

To alter the three variables in the Debugger, load it into one of the lower segments, say segment 4. Then go to the Wordease menu and select the editor, and target it to segment 4. Then globally change the four variables as follows:

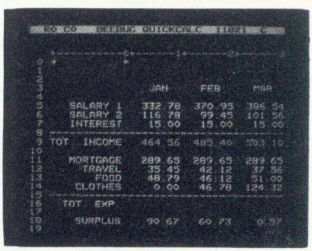
```
X% to R%
Y% to S%
Z% to T%
Z$ to X$
```

Now save segment 4 on your new Wordease disc under the filename W.N, and the job is done.

Once Wordease is booted up, the new entry will appear on the menu as selection "N", and it will load and run as any other Wordease utility. The only difference is that it also makes use of segment 7 to run the single-stepper, and there is no route back to the main Wordease menu from the Debugger menu, though the latter could be easily added.

Quickcalc — Tube compatible

This low cost Beebugsoft disc/cassette based spreadsheet, although not as powerful as some of its competitors on Rom, continues to be well received. It is fully 6502 tube compatible, utilising the extra memory for increased sheet size and has recently been reviewed favourably in Micro User (April 1986).



Magscan and Watford RAM

Mr D. Ridpath in Norway has sent us a hint to enable Magscan to be used with the Watford RAM Board. The problem was caused by the allocation of space to the RAM board workspace. The solution is to use the *MWS call to relocate this workspace as follows:

```
*BUILD !BOOT
0001*BASIC
0002*MWS 0A00
0003*RAMON
0004PAGE=&1500
0005CHAIN"INDEX"
0006<esc>
*OPT4,3
```

This enters Basic, sets the workspace at hex A00, turns on the RAM board (which is necessary as otherwise INDEX is too big) and then continues as in the supplied !BOOT file. <Shift><Break> then enters INDEX correctly.

COMPUTER SIMULATION

(Part 1)

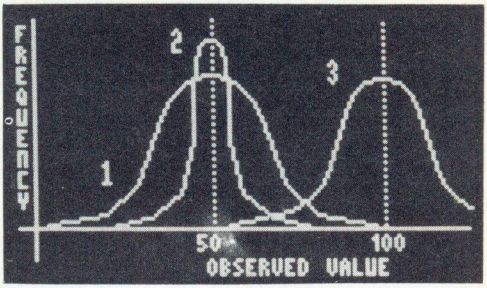
There is more to statistics than helping politicians out of tight corners. Careful use of simulation can help solve all kinds of problems in the real world. Jan Stuurman illustrates some useful computer techniques.

If you have ever recorded marks in a school register, helped with a census, or taken readings of any other kind, you've probably used some elementary statistics already. The simplest form of analysis is to work out the average (also known as the 'mean') of the figures obtained. You can analyse a set of numbers much more closely than this, however, and predict solutions to problems by studying statistical 'models' of them. The calculations involved in these simulations can be speeded up using a micro.

Any set of figures obtained from natural phenomena will be distributed across a range of possible values. Take, for instance, the pseudo-random number generator in BBC Basic (see also First Course, BEEBUG Vol.5 No.1). If you produce a thousand numbers in the range 1 to 100 using the function RND(100), you can expect the numbers to be distributed uniformly throughout the range. There should be roughly ten of each number between 1 and 100. Logically enough, this kind of distribution is said to be 'Uniform' and is ideal for simulating the throw of a die or the drawing of cards from a pack.

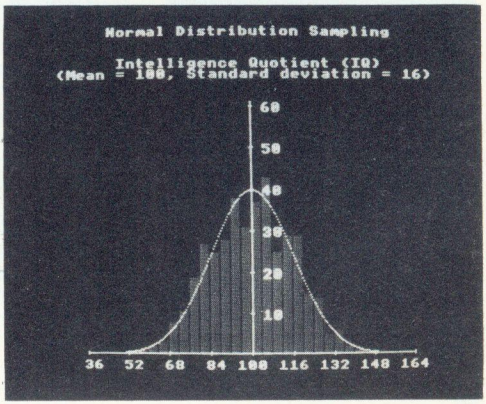
Sets of figures taken from real life, however, are more likely to fit a 'Normal' distribution. Normally distributed values tend to be clustered around their mean, with values smaller or larger than the mean being less frequent the further they are away from it. This produces a characteristic bell-shaped curve.

Examples of Normal distributions include the heights of people in a population, the lengths of leaves on a tree, the Intelligence Quotients (IQs) of a group of people and the weights of a set of ten pence coins. To model this kind of distribution on a micro, you need to be able to generate random numbers which also possess the Normal 'clustering' behaviour.



A normal distribution can be described by specifying its mean and standard deviation. The mean value of curves 1 and 2 is 50, while that of curve 3 is 100. A change of mean moves the whole curve left or right along the x axis. The standard deviation is effectively the 'spread' of numbers about the mean. Curves 1 and 2 both have the same mean, but curve 1 has a larger spread, or standard deviation.

The demonstration program simulates 400 observations of IQ in a group of people. IQ is a Normally distributed phenomenon which is known to have a mean of 100 and a standard deviation of 16. The program draws a bar graph by grouping random IQ values in 32 intervals, each 4 points wide. Once the bars have been drawn, a theoretical curve with the same parameters is superimposed on them for easy comparison.



FNnorm does most of the work in the simulation, calculating a Normal random number from the Uniform one available via the RND function. FNnorm can be used in your own programs to generate Normally distributed random numbers.

The bar graph and theoretical curve are centred around the mean and take values in the range 'mean-64' to 'mean+64'. 64 is four times the standard deviation and 99.99% of all numbers in the distribution should fall in this range.

AN EXAMPLE SIMULATION

Suppose that a company manufactures units consisting of three resistors linked as follows:

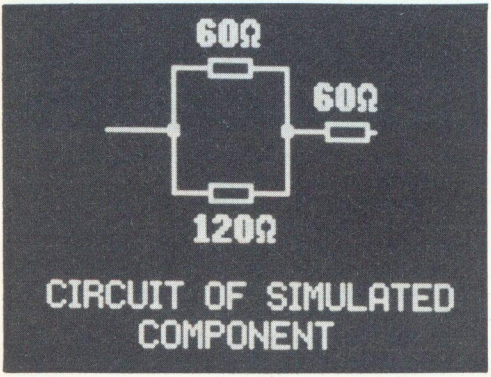


Diagram of resistor network

The total resistance of this unit is calculated as 1/(1/120+1/60)+60=100 Ohms. If the required tolerance of the unit is 2%, how will it affect the reject rate of units if the tolerance of each individual resistor is changed from 5% to 10%?

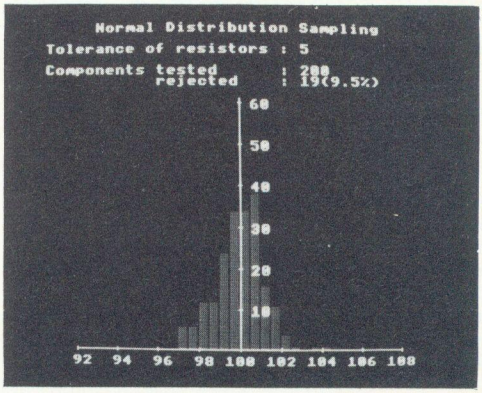
If you replace lines 1200 to 1460 of the original program with the second program segment, the program will then simulate this problem. To do so, it makes use of the fact that 99.8% of all Normal random numbers will lie within 3 standard deviations of the mean. So, for example, the 120 Ohm resistor with a 10% tolerance has a mean of 120 and an approximate standard deviation of (10% of 120)/3=4.

This is because with a tolerance of 10%, all resistors must come within the +/-10% band. To encompass all values, this band must be some 6 standard deviations wide (i.e. 3 either side of the mean). Thus the standard deviation of our idealised Normal distribution is 10% of 120 divided by 3, or 4 Ohms.

Run the program with 5% and 10% as the two resistor tolerances and you should see that lowering the tolerance of the resistors will increase the reject rate by roughly 500%.

Next month we'll look at why you have to wait so long in queues at the Post Office!

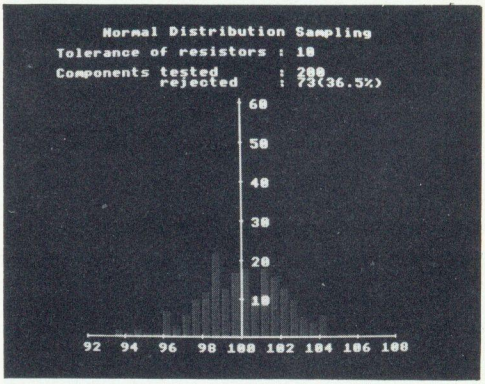
```
  10 REM PROGRAM NORMAL RANDOM NUMBERS
  20 REM Version B.1
  30 REM Author  Jan Stuurman
  40 REM BEEBUG June 1986
  50 REM Program subject to copyright
  60 :
 100 MODE1:VDU23;11,0;0;0;0;
 110 ON ERROR GOTO 2200
 120 PROCinit
 130 PROCdemo
 140 REPEAT UNTIL FALSE
 150 END
 160 :
1000 DEFPROCinit
1010 LOCAL I%:DIM bar%(31)
1020 CLS:PRINTTAB(6)"Normal Distributio
n Sampling"
1030 REM SET-UP GRAPH AREA & AXES
1040 VDU24,0;0;1279;799;
```




```
1050 VDU29,640;64;
1060 GCOL0,3
1070 MOVE-512,0:DRAW512,0
1080 MOVE0,0:DRAW0,799
1090 FOR I%=-512 TO 512 STEP 128
1100 MOVEI%,4:DRAWI%,-4
1110 NEXT I%
```

```
 1120 FOR I%=0 TO 720 STEP 120
 1130 MOVE4,I%:DRAW-4,I%
 1140 NEXT I%
 1150 ENDPROC
 1160 :
 1200 DEFPROCdemo
 1210 mean=100:stddev=16:samplesize=400
 1220 @%=3:VDU5:REM LABEL AXES
 1230 FOR I%=-4 TO 4:MOVEI%*128-48,-24:P
RINTmean+I%*stddev:NEXT I%
 1240 FOR I%=10 TO 60 STEP 10:MOVE0,I%*1
2+8:PRINTI%:NEXT I%:VDU4
 1250 PRINTTAB(7,3);"Intelligence Quotie
nt (IQ)"'" (Mean = 100, Standard deviati
on = 16)"
 1260 VDU 19,2,4,0,0,0:GCOL1,2
 1270 REM FIND RANDOM IQ-SCORES AND
         UPDATE BAR-GRAPH
 1280 FOR observation=1 TO samplesize
 1290 normalrnd=INT(FNnorm(mean,stddev,R
ND(1))+.5)
 1300 IF normalrnd<36 OR normalrnd>=164
THEN 1290:         REM OUTSIDE
 1310 PROCbargraph((normalrnd-36)DIV4)
 1320 NEXT observation
 1330 REM DRAW THEORETICAL NORMAL CURVE
 1340 PROCnormalcurve
 1350 ENDPROC
 1360 :
 1370 DEFPROCnormalcurve
 1380 LOCAL x,y:GCOL0,1
 1390 FOR x=36 TO 164 STEP.5
 1400 y=EXP(-((x-mean)^2)/(2*stddev^2))/
(stddev*SQR(2*PI))
 1410 PLOT69,(x-mean)*8,19200*y
 1420 NEXT x:ENDPROC
 1430 :
 2000 DEFFNnorm(MEAN,STDDEV,CDFVAL)
 2010 LOCAL sign,temp,norm
 2020 sign=1+2*(CDFVAL<.5)
 2030 temp=-LN(4*CDFVAL*(1-CDFVAL))
 2040 norm=sign*SQR(temp*(2.0611786-5.72
62204/(temp+11.640595)))
 2050 =norm*STDDEV+MEAN
 2060 :
 2100 DEFPROCbargraph(interval%)
 2110 IF interval%<0 OR interval%>31 THE
N ENDPROC
 2120 LOCAL I%
 2130 FOR I%=3 TO 9 STEP 3
 2140 MOVE32*(interval%-16),bar%(interva
l%)+I%:PLOT1,24,0
 2150 NEXT I%
 2160 bar%(interval%)=bar%(interval%)+12
 2170 ENDPROC
 2180 :
 2200 ON ERROR OFF
 2210 MODE7:VDU23;11,255;0;0;0;
 2220 IF ERR=17 END
 2230 REPORT:PRINT" at line ";ERL
```

```
 1200 DEFPROCdemo
 1210 samplesize=200
 1220 @%=3:VDU5:REM LABEL AXES
 1230 FOR I%=-4 TO 4:MOVE I%*128-48,-24:
PRINT100+I%*2:NEXT I%
```




```
 1240 FOR I%=10 TO 60 STEP 10:MOVE 0,I%*
12+8:PRINTI%:NEXT I%:VDU4
 1250 PRINTTAB(1,2);:INPUT"Tolerance of
resistors : "tolerance
 1260 VDU19,2,4,0,0,0:GCOL1,2
 1270 reject=0
 1280 REM SIMULATE CONSTRUCTION OF COMPO
NENTS
 1290 FOR component=1 TO 200
 1300 resistor1=FNnorm(120,1.2*tolerance
/3,RND(1))
 1310 resistor2=FNnorm( 60,0.6*tolerance
/3,RND(1))
 1320 resistor3=FNnorm( 60,0.6*tolerance
/3,RND(1))
 1330 resistance=1/(1/resistor1+1/resist
or2)+resistor3
 1340 IF resistance<98 OR resistance>102
 THEN reject=reject+1
 1350 PROCbargraph(INT((resistance-92)*2
))
 1360 NEXT component
 1370 PROCresults
 1380 ENDPROC
 1390 :
 1400 DEFPROCresults
 1410 PRINTTAB(1,4)"Components tested";S
PC6;": ";samplesize
 1420 PRINTTAB(12,5)"rejected    : ";rej
ect;"(";INT(reject/samplesize*1000/10);"
%)"
 1430 ENDPROC
 1440 :
```

SUPERCHARGING THE MASTER

If you are seeking the ultimate super fast machine, the Turbo upgrade for the Master 128 could be the answer. Peter Rochford has been trying to keep up with this latest wonder.

With the launch of the new Master 128, Acorn have again provided a base machine that will satisfy the needs of many users, but to which you can add a choice of co-processor for those who need the extra power and facilities.

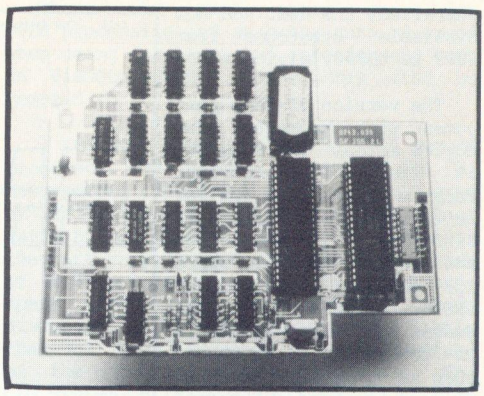
The Model B allows the use of an Acorn 6502 or Z80 second processor and those produced by independent manufacturers such as Torch. All of these are external and connect to the host computer via the Tube socket. The Master allows the same, but in addition you can add an 8, 16, or 32 bit internally mounted co-processor.

The 8 bit co-processor has been dubbed the 'Turbo' by Acorn and retails for £125 as opposed to £199 for the old external 6502. It can be supplied already fitted to the Master or, as an upgrade kit comprising the Turbo board, six support pillars, a rather thin manual (13 pages) and a support disc.

INSTALLATION

Fitting the 6" x 6" Turbo board is simplicity itself and takes about five minutes. The ends of the board have very substantial rows of pins that plug into two sockets on the Master's main PCB. The addition of the support pillars gives you a very secure and stable installation, with no leads to connect or links to cut.

The Turbo boasts some 21 chips and included amongst these are 64K of fast RAM, a boot ROM, Tube ULA and a CMOS 65C102 microprocessor. The standard of construction, as one would hope from Acorn, is excellent. Once installed and with the lid back on the computer, you don't know the Turbo is there. Far better that the old 6502 second processor that



took up so much desk space and had a ridiculously short and uncooperative ribbon cable.

Powering up the computer reveals no change in the start up message until you realise that the Master at this point does not know that the new addition exists. Two commands must now be entered. *CONFIGURE INTUBE to tell the Master to select the internal co-processor and a *CONFIGURE TUBE to make it active. The reverse of these is *CONFIGURE EXTUBE to select an external co-processor such as a 6502 or Z80 connected to the Tube socket and a *CONFIGURE NOTUBE to switch off either the internal or external processor depending on which is selected. All these parameters are stored by the Master's battery-backed RAM. Thus once you have re-configured your Master, it will always power up with the co-processor operational (similarly when pressing Ctrl-Break).

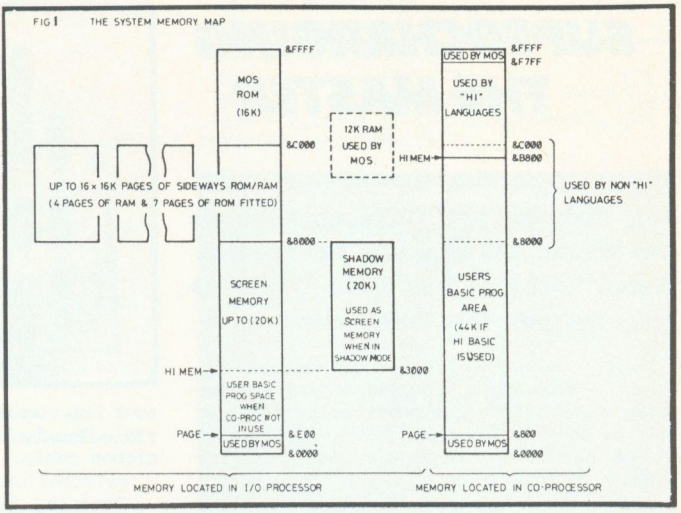
MEMORY CONFIGURATION

With the Turbo selected, the message 'Acorn Tube 65C102 Co-Processor' is displayed. At this point the current language, i.e. Basic, is copied across the Tube and located at &8000 in the Turbo. Looking at Fig.1 will best illustrate the memory mapping of the Master and the Turbo. With the current language copied across to the co-processor the user has 30.7K of memory to play with. However, because the co-processor shares the MOS with the I/O processor the area from &C000 to &FFFF is vacant. To take advantage of this, Acorn have provided a version of Basic on the Turbo support disc called HI-Basic, which is assembled to run at &B800. This means that with HI-Basic

installed, the user RAM now available stretches from &800 to &B800, giving 44K.

The version of View that comes with the Master also copies across the Tube but it is self-relocating. Because View is a smaller language than HI-Basic, it sits higher in the Turbo and leaves 48.3K of space for text. HI versions of various other ROM languages are also available to run in the co-processor such as Computer Concepts' HI-Wordwise and HI-Sheet. HI-Edit, a version of the Master's text editor, is supplied on the Turbo support disc.

FIG 1 THE SYSTEM MEMORY MAP



Any ROM that is written to Acorn's rules should work across the Tube and run with the co-processor. A full list of Tube compatible ROMs has been published in past issues of BEEBUG (last update in Vol.5 No.1). As for other software, provided it follows Acorn's rules there is no reason why it should not work.

THE TURBO IN USE

So having got the Turbo and installed it, what can it be used for and what are the advantages? Well, we have already mentioned the extra memory available in View, for Basic, and other HI languages. As the name implies, the Turbo provides something else - SPEED! The Turbo runs at a clock rate of 4MHz compared to the 2MHz of the Master and Model B, whilst the old 6502 second processor runs at a clock rate of 3Mhz. The 4MHz 65C102 microprocessor is the reason why the Turbo is equipped with 64K of FAST RAM.

Take a look at the table in Fig.2. You will see that the Master is already a very fast machine on the PCW Basic benchmark tests. It is three times faster than the standard Model B and slightly faster than the Model B with the old 6502 second processor. With the Turbo connected, the Master turns in a performance that puts many of the new 16 bit micros to shame. For example, the Master Turbo is around four times faster than an IBM PC on the PCW benchmarks.

Don't be deceived though, Basic benchmarks are all right to give an indication of the differences in speed when the computer is set certain tasks. In reality, when running normal programs things can look quite different. Ones that require a lot of calculation and writing to the screen, such as for complex graphics, will show very significant increases in speed with a Turbo. On the other hand, a program that contains many timing loops or constantly accesses the filing system will not benefit much from the Turbo in terms of execution time.

Running spreadsheets, databases and wordprocessors with the Turbo all show varying degrees of gain in speed, and a small increase in available space. With a spreadsheet such as ViewSheet, on the small tests that I performed, there were increases in speed up to 50% when recalculating Using the excellent ViewStore database, I found that in general use, there is little gain in speed with the Turbo apart from faster screen scrolling. When performing any task using the utilities however, there was a marked increase due to the fact that the utilities reside in the I/O processor and do not occupy any of the Turbo RAM. This leads to less disc accessing during sorting and indexing.

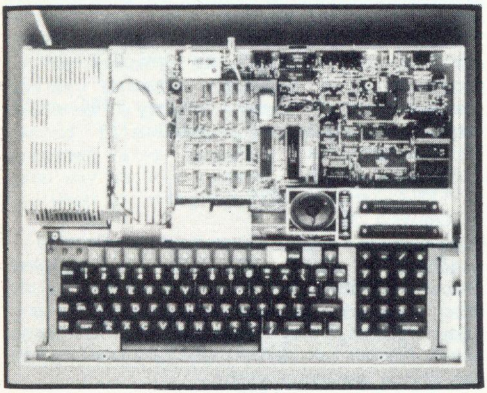
When using the wordprocessor View, the only really significant increase in speed is in the screen scrolling. With Wordwise

Plus there did not appear to be much of a difference at all when using the Turbo. On the Turbo support disc by the way, is a printer buffer utility that configures 24K of the I/O processor memory as an extended buffer. This is a most useful feature, particularly if you have a comparatively slow printer such as a daisy-wheel.

If you are a games fan, the bad news is that to my knowledge there are few if any commercial games written to run using a co-processor. I believe Acornsoft are to release a version of Elite that utilises the co-processor and features full colour graphics and extra speed but I have yet to see it. Mind you, the mind boggles when one considers the possibilities for games written for the Turbo with its speed and available RAM.

CONCLUSION

With such an increase in space and speed, the Turbo provides some tremendous possibilities for the serious programmer, particularly for large and complex graphics programs. I can also see it being greeted with enthusism by schools and universities for specialist applications.



Those who want extra space for word processing and an extra turn of speed when using databases and spreadsheets such as ViewStore and ViewSheet, may also be tempted to invest in one.

I remember though, when the 6502 second processor was launched for the Model B. Many people rushed out and bought one even at its price of £200. I have spoken to a large number of 6502 owners who say that they are very disappointed with the lack of software support for it from Acorn. Apart from the applications already discussed in this review, there is little that has been released to take advantage of the potential of a co-processor. The CAD system BITSTIK was written to utilise a co-processor but that is of no use to the average user. Most of the old 6502 co-processors in existence at the present I feel must lie idle a lot of the time.

The Turbo scores over the old 6502 apart from speed and its better way of connection to the host computer, in one other area - its price. At £125 it is much cheaper and this fact could make it far more successful than its predecessor by encouraging greater sales. If this happens then we may see much more software being written to take advantage of its enormous potential. Conversely, if Acorn or an independent software house released a program of wide appeal that took full advantage of the speed and space of the co-processors, I feel that would be enough to encourage many Master owners to dash out and buy the Turbo at £125. Perhaps this will happen with the promised new version of Elite.

Personally, I am most impressed and excited by the possibilities of the Turbo and really hope that it receives the software support it needs and deserves.

PCW BASIC BENCHMARKS (Eight tests averaged out)

Computer	Processor	BASIC	RAM	Time (secs)
Master & Turbo	65C102	HI-BASIC 4	44K	4.43
Master	65C12	BASIC 4	28K	9.37
Model B & 6502 SP	6502B	HI-BASIC 2	44K	10.38
Model B	6502A	BASIC 2	25K	15.1

Note: Timings will vary slightly from machine to machine.

Figure 2

Beebug Filer goes Graphic

Mike Williams adds an option to the BEEBUG Filer, our data base program, providing a comprehensive and flexible graphical display for your data.

BEEBUG Filer is a database management system described in BEEBUG Vol.4 Nos.6 to 8. We have already provided one enhancement to the original program in the form of a mail-merge facility (described last month). We now present a completely new and separate program that will allow the contents of the datafiles you have created with Filer to be displayed graphically. This new program allows you considerable freedom in choosing an appropriate graph for your data, and includes several useful and practical features.

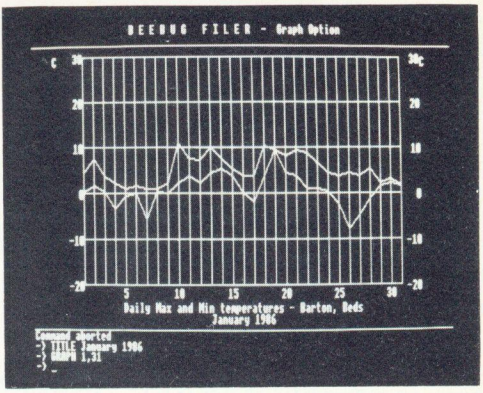
THE FILER GRAPH PROGRAM

The program listed with this article (FILERG) follows the same general structure as the main Filer program (indeed several of the same procedures are incorporated in it), and it is command driven as before. The program uses mode 0, and must be run with PAGE set to &1400 (unless using a Master 128 or a B+ with shadow memory).

To describe the layout of a graph, the Filer Graph program uses the idea of a 'graph format' file, very similar in concept to the 'format' file used to control printing in Filer. The graph format contains all the information about axes and scales.

The commands recognised by the Filer Graph program are shown below.

OPEN	Open a data file
GFORMAT	Select a graph format
GRAPH	Draw a graph
TITLE	Specify title for graph
PRINT	Produce copy on printer
COMMANDS	List these commands
END	Exit from this program
* command	Execute any * command



PRODUCING A GRAPH

Let us suppose that we are using Filer to maintain a database containing the weather data comprising daily maximum and minimum temperatures for 1986. The datafile is called TEMPS86 and contains, among other information, the two fields TMAX and TMIN. Assuming that we have already created suitable graph formats, producing a graph showing daily temperatures for, say, January 1986 would require the following commands:

```
OPEN TEMPS86
GFORMAT GRAPH1
TITLE January 1986
GRAPH 1,31
```

Given a suitable format file this will produce a graph of daily temperatures throughout the month of January. If we also required a similar graph for any other month, say February, then the same graph format could be used by continuing the above sequence with:

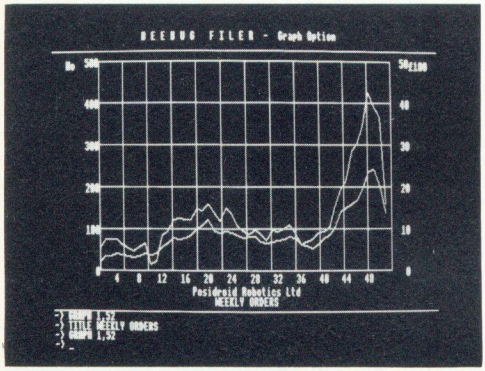
```
TITLE February 1986
GRAPH 32,59
```

The use of these commands is straightforward. GFORMAT selects a graph format, TITLE specifies any title or heading to be placed under the graph, while the GRAPH command actually initiates the drawing of the graph by giving the numbers of the first and last records to be included. The results can be seen in the illustrations.

If the GRAPH command is used without any record numbers then just the axes and scales are displayed. This is useful for checking that a graph format is correct before actually plotting the data.

HARD COPY OUTPUT

The PRINT command allows you to output any graph to a printer. The listing (at line 1670) contains the appropriate command for Printmaster. Change this to the version for your printer dump ROM, or to a call to your own printer routine.



GRAPH FORMATS

The use of graph formats is central to the way in which the Filer Graph program produces a graph. A graph format is a one record file that you create with the original Filer program, very much in the same way as you create print formats for use when printing out records.

As listed, the Filer Graph program can display up to four data fields on the same graph. This could be increased, if you have sufficient memory, by changing the sizes of the arrays (var$, Min, Max, label$, Int) at line 1060.

Using BEEBUG Filer, you could create a graph format for use with the file TEMPS86 by proceeding as follows:

```
CREATE GRAPH1/2
OPEN GRAPH1
ADD
Confirm (Y/N): Y
END
```

The special form of the CREATE command used above will create a file of the right size with the name GRAPH1 (or whatever you choose). The value 2 indicates that two data fields are to be displayed in this example. We then open that file and add one record, the graph format, to it, confirm that the entry is correct and exit from Filer. If we have made no mistakes, we can now use this graph format with the Filer Graph program to produce a graph.

Let us now look at the graph format itself. The record that we create will consist of a header line followed by one line for each of the data fields to appear on the graph.

THE HORIZONTAL AXIS

Taking the header line first, this consists of a list of numbers, followed optionally by some text, in the order:

Min,Max,n1,n2,n3,n4/text

The numbers to be specified here are:

Min	Minimum x value (usually 0 or 1)
Max	Maximum x value
n1	Number of data points to be plotted
n2	Number of vertical lines (min 2)
n3	First scale value
n4	Interval between scale values

The optional text provides a label or heading displayed below the x axis.

All the information in this first or header line describes the x (horizontal) axis of the graph. The number of data points is the number of records that will be read to create the graph. To make reading the resulting graph easier you can specify any number of evenly spaced vertical lines to be drawn. You should always specify at least two, representing the left and right hand vertical axes.

To mark off the the horizontal scale, you should specify the scale value of the first point to be so marked, and the interval between subsequent scale values. It is often advisable not to label the origin as any overlap of vertical and horizontal scale values here can be confusing to read.

These numbers should be separated by commas as shown. If the optional text is included, then this is preceded by a '/' character, otherwise the '/' is omitted.

THE VERTICAL AXES

Each subsequent line of the graph format deals with a data field and its associated vertical scale. Note that although up to four data fields may be plotted, only two vertical scales are possible. The format of each such line is:

<data field>,Min,Max,label/conversion

The name of the data field should be specified first, followed by the minimum and maximum values for the corresponding vertical scale and a label for that scale (not more than five characters). If the scale parameters are omitted then those for the previous data field will be used. The very first data field must have a scale specified though.

The conversion string, and preceding '/' character, is optional but may be used whether a scale is included or not. This allows values as recorded in a datafile to be converted before being displayed on the graph. In the example above, TMAX and TMIN might be recorded in Fahrenheit but could be displayed in Celsius by including conversion factors of:

```
(TMAX-32)*5/9
(TMIN-32)*5/9
```

SOME EXAMPLES

The graph format GRAPH1, referred to previously, was thus created as:

```
1,31,31,31,5,5/Max and Min Temperatures
TMAX,-20,30,C/(TMAX-32)*5/9
TMIN/(TMIN-32)*5/9
```

By specifying the right set of records with the GRAPH command, this one graph format could be used to produce a graph of daily temperatures for any month in the year. As another example, the following graph format will allow three months of data (90 data points) to be displayed on the one graph:

```
1,91,91,4,5,5/Max and Min Temperatures
TMAX,-20,30,C/(TMAX-32)*5/9
TMIN/(TMIN-32)*5/9
```

Although graph formats, like print formats, may seem complicated at first sight, time spent experimenting and getting to know their capabilities will be well worth while. We have also considered but one example; there are many others (sales figures, share prices, bank balance for example).

FURTHER NOTES ON GRAPH FORMATS

In general, you are likely to get the best and most accurate results by choosing ranges for horizontal and vertical scales which are multiples of 10. The Graph program will also cater for any missing values, if these are represented in your datafile by the value -9999. When the graph is drawn, any such values will be ignored and the graph drawn from the last 'good' point to the next 'good' point using a dotted line (though this feature may not always be very apparent.

PROGRAM NOTES

The Filer Graph program uses several procedures from the main Filer program. These should be copied from your version of Filer and added to the program listed below. The procedures used this way are:

PROCtitle	PROCinv
FNvalidate	PROCread
PROCopen	PROCstrip
PROClistc	PROCstar
PROCwindow1	PROCoscli
PROCwindow2	

Once this is done you will also need to change 3 of the original lines as follows:

```
3260F$=p$:F=OPENUP(F$):graph%=0
3320FORI=1TOf:INPUT#F,p$,p%:field$(I)=FNstrip(p$,"."):NEXT
6320PROCwindow1:CLG
```

Memory space is extremely tight with this program. When typing the program in, do not add any extra spaces and do omit the lines with just REM statements or colons. Leave out also the space between the line number and the start of the program. These have been left in the listing to aid readability. If you have a program compactor (as in BEEBUGSOFT's Toolkit) then use this as well to produce a working version of the program.

At all times, PAGE must be reduced to &1400 before loading and running this program, though on the B+ and Master 128, the use of shadow RAM can avoid all these potential problems.

Copies of the notes for the main Filer program are still available on receipt of an A5 SAE to the St Albans address.

```
  10 REM BEEBUG FILER GRAPH B1.0
  20 REM Author Mike Williams
 100 MODE0:GCOL0,1:VDU26:PROCsetup
 120 PROCtitle:PROCheader:ONERRORPROCerror
 140 REPEAT:PROCcommand:UNTILexit%
 160 PROCclose:VDU26:CLS:*FX4,0
 180 END
 200 :
1000 DEF PROCsetup
1020 LOCAL I:exit%=0:open%=0:graph%=0:VDU24,0;136;1279;924;
1040 maxf=12:X=0:Y=0:w=1:FDR=256:Title$=""":xlabel$="":*FX4,2
```

```
 1050 xscale=960:yscale=640:VDU29,160;256;
 1060 DIMrecord$(maxf,1),field$(maxf),xd
ata(5),var$(3),Min(3),Max(3),label$(3),I
nt(3),os 40
 1080 READ N:DIMcom$(N)
 1100 DATA 8
 1120 FORI=1TO N:READcom$(I):NEXT
 1140 DATA OPEN,CLOSE,GFORMAT,GRAPH,PRIN
T
 1160 DATA COMMANDS,TITLE,END
 1180 ENDPROC
 1200 :
 1500 DEF PROCcommand
 1520 LOCAL command$,pm$
 1540 REPEAT
 1560 INPUT LINE"-> " command$
 1580 C=FNvalidate(command$)
 1600 IF C=0 PRINT"Unrecognised command"
 1620 UNTIL C
 1630 IF C=1 PROCopen(pm$)
 1640 IF C=2 PROCclose
 1650 IF C=3 PROCformat(pm$)
 1660 IF C=4 PROCgraph(pm$)
 1670 IF C=5 THEN *GDUMP
 1680 IF C=6 PROClistc
 1690 IF C=7 Title$=pm$
 1700 IF C=8 exit%=TRUE
 1870 IF C=99 PROCstar(pm$)
 1880 ENDPROC
 1890 :
 6000 DEF PROCclose
 6020 IF NOT open% ENDPROC
 6040 PRINT"File closed - ";rec-1;" reco
rds in use"
 6060 CLOSE#F:open%=0:CLG
 6080 ENDPROC
 6100 :
 6500 DEF PROCformat(p$):LOCAL F,I,J,p,n
ame$,var$
 6520 IF NOT open% PRINT"No file open":E
NDPROC
 6540 IFp$="" PRINT"No file given":ENDPR
 6560 F=OPENUP(p$):IF F=0 PRINT"No such
file":ENDPROC
 6580 INPUT#F,frec,I,I,ff:IFfrec=1 PRINT
"No graph format":ENDPROC
 6600 PRINT"Loading graph format ";p$:F1
$=p$
 6620 PROCwindow1:PROChead(19,0,"GFormat
: "+F1$,0,0,0):PROCwindow2
 6640 PTR#F=FDR:FOR I=0 TO ff-1:INPUT#F,
p$:record$(I,1)=FNstrip(p$,"."):NEXT I
 6660 p$=record$(0,1):p=INSTR(p$,"/"):IF
p xlabel$=MID$(p$,p+1):p$=LEFT$(p$,p-1)
 6680 p$=p$+",":I=0:REPEAT:p=INSTR(p$,",
"):xdata(I)=VAL(LEFT$(p$,p-1)):p$=MID$(p
$,p+1):I=I+1:UNTIL p$=""
 6700 FORI=0TOff-2:p$=record$(I+1,1):p=I
NSTR(p$,"/")
 6720 IFp var$=MID$(p$,p+1):p$=LEFT$(p$,
p-1) ELSEvar$=""
 6740 p$=p$+",":J=1:REPEAT:p=INSTR(p$,",")
 6760 IFJ=1 var$(I)="VAL(record$("+FNfie
ld(LEFT$(p$,p-1))+",0))":name$=LEFT$(p$,
p-1)
 6780 IFJ=2 Min(I)=VAL(LEFT$(p$,p-1))
 6800 IFJ=3 Max(I)=VAL(LEFT$(p$,p-1))
 6820 IFJ=4 label$(I)=LEFT$(p$,p-1):Int(
I)=Max(I)-Min(I):A=I
 6840 J=J+1:p$=MID$(p$,p+1)
 6860 UNTILp$=""
 6880 IFJ<5 Min(I)=Min(I-1):Max(I)=Max(I
-1):label$(I)=label$(I-1):Int(I)=Int(I-1
)
 6900 IFvar$<>"" p=INSTR(var$,name$):var
$(I)=LEFT$(var$,p-1)+var$(I)+MID$(var$,p
+LEN(name$))
 6920 NEXT:graph%=-1:CLOSE#F:ENDPROC
 6940 :
11000 DEF PROCgraph(p$):LOCAL p,p1,p2
11020 IF NOT graph% PRINT"No graph forma
t":ENDPROC
11060 s1=xscale/(xdata(2)-1):s2=xscale/(
xdata(3)-1):s3=(xdata(4)-xdata(0))*s1:s4
=xdata(5)*s1
11080 PROCgraph1:IFp$="" ENDPROC
11100 p=INSTR(p$,","):IFp=0 PRINT"Bad li
mits":ENDPROC
11120 p1=VAL(LEFT$(p$,p-1)):p2=VAL(MID$(
p$,p+1))
11140 IFp1<1 OR p2>=rec PRINT"Bad limits
":ENDPROC
11160 IFrec<2 PRINT"No records in file":
ENDPROC
11180 PROCgraph2(p1,p2)
11200 ENDPROC
11220 :
12000 DEF PROCgraph1:LOCAL n,x,y:CLG:VDU
5:n=xdata(4)
12020 FORx=0TOxscale STEPs2:MOVEx,0:DRAW
x,yscale:NEXT
12040 FORy=0TOyscale STEP128:MOVE0,y:DRA
Wxscale,y:NEXT
12060 FORx=s3 TOxscale STEPs4:v$=STR$(n)
:MOVEx-LEN(v$)*8,-12:PRINTv$:n=n+xdata(5
):NEXT
12080 FORy=0TOyscale STEP128:y$=STR$(Min
(0)+y*Int(0)/yscale):MOVE-LEN(y$)*16,y+8
:PRINTy$:NEXT
12100 FORy=0TOyscale STEP128:y$=STR$(Min
(A)+y*Int(A)/yscale):MOVExscale+16*(4-LE
N(y$)),y+8:PRINTy$:NEXT
12120 MOVE-80-16*LEN(label$(0)),yscale:P
RINTlabel$(0):MOVExscale+80,yscale:PRINT
label$(A)
12140 MOVE(xscale-16*LEN(xlabel$))/2,-52
:PRINTxlabel$
12160 MOVE(xscale-16*LEN(Title$))/2,-84:
PRINTTitle$
```

→18

WIGMORE MOUSE

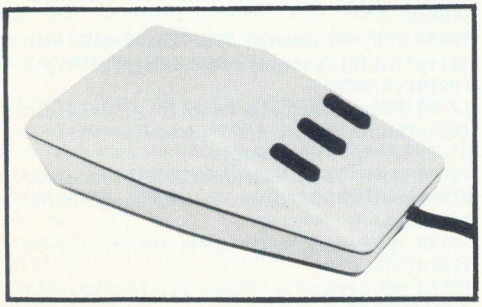
Terry Hallard takes an enthusiastic look at the latest mouse and software, this time from Wigmore House. Is this set to give AMX a run for its money?

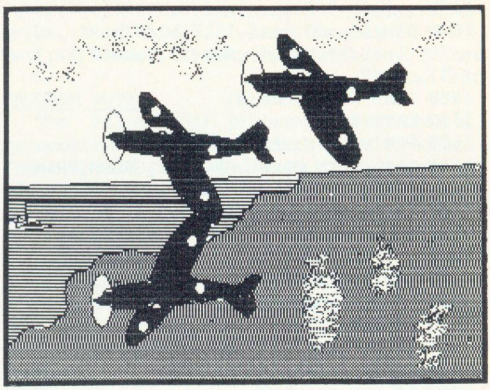
Products : Megamouse £68.89
The Artist £57.39
(both together £114.89)
Cadmouse £34.39
Supplier : Wigmore House Ltd
32, Saville Row,
London, W1X 1AG.

Unobserved by many people, a smart new mouse has recently entered the BBC market place. Its producers are very modest and do not seem to be mounting an attack upon AMX, rather the opposite. Wigmore House Ltd are taking a distinctly quiet approach with their Megamouse and its two accompanying software packages.

MEGAMOUSE

The design of the mouse is very different from the familiar chunky black AMX product. It actually LOOKS like a mouse - long, grey and sleek. Ergonomically designed, it fits the hand beautifully, with no cramped feeling. The buttons on the wedge-shaped front end fall exactly under the fingers. The ball, unlike the polished steel of its cousin,





is made of hard black rubber (the latest AMX mouse is a new design which also uses a similar black rubber ball) which will not slip on any surface - even glass!

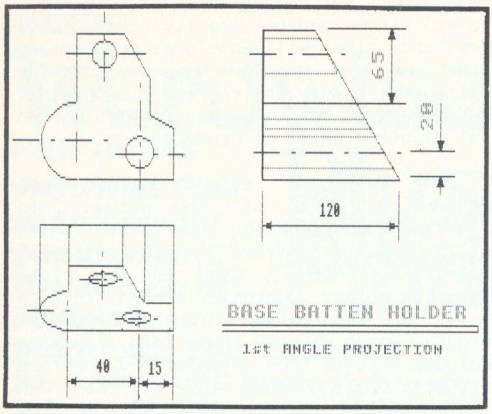
The mouse, attached to the Beeb by a metre of cable, glides around smoothly and works with all the AMX products that I have tried, except the original AMX Art package.

CADMOUSE

There are two accompanying software packages at present available from Wigmore. Cadmouse comes on disc and is geared principally for the Acorn DFS. An A5-sized 26-page handbook is supplied, with diagrams dumped from the program itself onto an Epson dot-matrix printer, which is the main type of printer that the program supports. Machine code dumps for other printers can be added by the user. The directions and explanations given are clear and readable, and the program is quite comprehensive.

It is similar to many drawing programs available for the Beeb but has many ambitious extras. The working screen has two menus, an options menu consisting of boxed icons up the right hand side of the screen, and a similar command menu along the bottom. This leaves a substantial area of the screen for the user.

All the expected drawing features are there (using 4-colour mode 1 or finer, 2-colour, mode 0): the usual lines (freehand, dotted and rubber-banded straight ones) plus a 'corrected line' (which straightens out lines where required, either horizontally or at 30'



angular increments, automatically correcting errors in placement), an excellent 'centre line', and a sophisticated 'dimension line' function. There is a choice of open or filled geometrical plane figures - rectangle, circle or arc - and a polygon call which allows shapes of up to 8 sides, or complete ellipses. The technical drawing illustrating this review makes extensive use of these functions.

There is a 'fill' option, whose textures can be complemented by 'palette' changes and extended by other commands. 'Paint' allows the user to choose from a large number of brush widths and types or even an airbrush spray, while 'hatch' gives a wide choice of different filling patterns including 45 degree hatching and various useful combinations of dots and stripes. A couple of these can be seen in the aircraft picture. The screen dump, incidentally, is one of the fastest that I have come across.

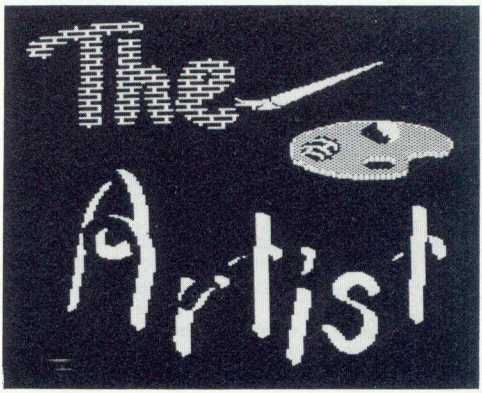
The only real criticism that I have is the absence of an 'eraser' function such as in the AMX packages. Instead, this means creating a special 'black filled rectangle' every time you want to remove a clanger (although immediately after an item has been drawn, pressing the right hand 'cancel' button will delete it).

There are also three useful and quite sophisticated functions which rely on a window being created around a feature - in this instance the first aeroplane, which was drawn freehand. It was then 'dragged' around the screen until the best position was found. Then it was 'duplicated' to form a pair and then one more was 'Inverted' to give a wing-over attacking effect. All in all, I would say that this is a very useful CAD package, using the Beeb's graphics capabilities to the full.

THE ARTIST

Nice as Cadmouse is, it is 'The Artist' which is the real stunner. I don't know how many readers have seen on TV the mainframe Megabyte-gobbling package which allows cartoon film designers to create their products rapidly, especially backgrounds. The designer only has to define a flower once and he can 'spray' it around where needed. Other effects allow multiple branched trees to be quickly brushed in as can long freehand lines made up of adjoining lengths of different colours. Well 'The Artist' not only lets the user do these things easily and more, it also lets you create simple animated pictures.

The first demonstration I saw of this package swiftly put a waterfall on the screen - then the water started to gush over the fall, rippling and splashing in the pool beneath! Another screen showed the black outline of a Frankenstein castle and a craggy path leading to it - then irregular flashes of lightning lit up the whole picture, showing detail and colour, only to plunge back into dark shadow again until the next flash! The effect is impressive, though difficult to describe adequately. You need to see it for yourself.



The package contains a 16k ROM, utilities disc and handbook. The manual adopts a quite lighthearted approach

which, while giving a neat, one page explanation of each function, encourages the user to explore for himself the goodies on offer. Examples are given of how to obtain some special effects, but in the main you can get great fun out of just 'fiddling around' and achieving results by accident.

The Artist is menu driven, the main menu offering colour selection and a number of manipulating options which are called 'flags'. The commands range from the usual line, triangle, rectangle, circle and ellipse drawing, through to 'fill' and many special effects.

The flags are the real key to the program's versatility, allowing the most subtle and ingenious effects to be achieved. For example, areas of the screen can be repeatedly copied anywhere, normal, reversed or upside-down. However, copying a rectangle usually produces unwanted 'corner bits', especially if copies are superimposed. Use of 'without' does away with all this. Using the 'special effects' menu, you can indicate any number of colours which will be ignored when copying. Thus if we have a shape to copy, other than a rectangle, 'without' the background colour allows only the shape to be copied and placed elsewhere. Similarly, 'over' allows you to specify that, say, red and blue will not overlay white or green on the screen. Now any drawing option will cause the red and blue BUT NOT ANY OTHER COLOUR to appear to pass behind any existing white or green object.

The 'animation' effect depends upon converting part of the screen into flashing colour. First the areas which will move are drawn in the 'static' colour, but with a mind to the right flashing colour eventually needed. This is saved and then passed through a 'convert' program which changes them into flashing colours. It is then reloaded and the rest of the drawing put in - then an 'animation' program is appended. On booting up, the picture is drawn statically and then the flashing colours take over giving a very effective display.

I have by no means covered all the possible effects, partly because I am discovering new techniques each time I use the package. All I can say is that I thoroughly recommend this program to anyone wishing to get some really stunning graphic output from the Beeb. Indeed, Wigmore can feel pleased with themselves that they have, like AMX, greatly enhanced the capabilities of the BBC micro.

←15

FILER GRAPH

```
12180 VDU4:ENDPROC
12200 :
13000 DEF PROCgraph2(p1,p2):LOCAL k,p,x,
y,oldx,oldy,I:k=0
13020 FORI=0TOff-2:p=p1
13040 PROCread(p,0):oldy=FNeval(I):oldx=
0
13060 FORx=s1 TOxscale STEPs1:p=p+1
13080 PROCread(p,0):y=FNeval(I)
13100 IFoldy<0 oldx=x:oldy=y ELSE IFy>0
MOVEoldx,oldy:PLOT5+k,x,y:oldx=x:oldy=y:
k=0 ELSEk=16
13120 IFp>=p2 x=1000
13140 NEXTx,I:ENDPROC
13160 :
13180 DEF FNeval(I)=(EVAL(var$(I))-Min(I
))*yscale/Int(I)
13200 :
20200 DEF PROCheader
20220 PROCwindow1:CLG
20240 IF open% THEN PROChead(0,0,STRING$
(80," "),0,0,0):PROChead(1,0,"File: "+F$
,0,0,0):PROChead(39,0,"Number of records
: ",rec-1,4,-1)
20250 IF graph% PROChead(19,0,"GFormat:
"+F1$,0,0,0)
20260 PROCwindow2:ENDPROC
20280 :
21000 DEF PROChead(x,y,msg$,n,w,flag)
21020 PROCinv(1):PRINTTAB(x,y)msg$;
21040 IF flag PRINT SPC(w-LEN(STR$(n)));
n
21060 PROCinv(0):ENDPROC
21080 :
21600 DEF FNfield(p$):LOCAL I:I=0
21620 REPEAT:I=I+1:UNTIL p$=field$(I) OR
 I>f
21640 =STR$(I)
21660 :
31000 DEF PROCerror
31010 IF ERR=17 THEN PROCheader:PRINT'"C
ommand aborted":ENDPROC
31012 IF ERR>189 AND ERR<208 REPORT:PRIN
T:ENDPROC
31020 ON ERROR OFF:PROCwindow2:REPORT:PR
INT" at ";ERL
31030 PROCclose:VDU26:*FX4,0
31040 END
```

DEBUG Wordwise PROGRAMS Plus Plus Plus

Wordwise Plus is deservedly popular, as is the writing of segment programs. This utility by P.L. Owen will come in handy when it's debugging time.

INTRODUCTION

The utility described in this article is a debugging tool for the Wordwise Plus programming language. It allows you to single-step through any Wordwise Plus program that you are testing, continually displaying the line to be executed next, the updated Wordwise segment and program screens, and the values of selected variables.

In this respect it behaves similarly to BEEBUGSOFT's Sleuth, though it lacks many of the sophistications of that Basic debugger. Like Sleuth, this utility allows you to switch between displays of current program line and the values of selected variables, and a display screen.

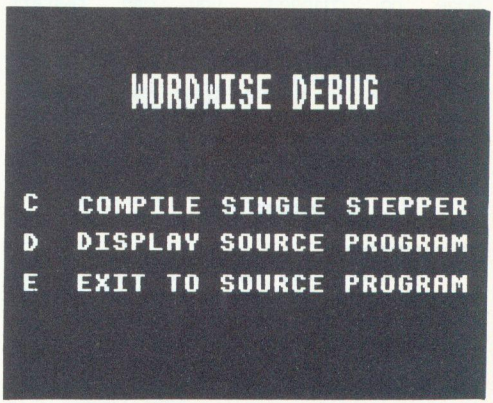
In Wordwise Plus there are two display screens of interest: the program display screen, showing anything that your program may have printed, and the currently selected segment on which your program may be working.

This debugger allows you to observe both. During operation, a first press of the space bar gives a readout of selected variables, and prints the next program line to be executed, each line being prefaced with a reference line number. The second press shows the current program screen after execution of that line, and if that instruction prints no text to the screen, you will see no change at this point, though the debugger issues a beep at each press as an audible cue. The third press of the space-bar reveals the currently selected segment, and again you will see no change here unless the last program line acted upon the segment in some way. A fourth press will take you full circle, displaying the next line to be executed together with selected variables, and so on.

USING THE DEBUGGER

To use the debugger, first type the program called DEBUGGER, listed below, into segment 6 of Wordwise Plus, but do NOT type in the accompanying line numbers which are for reference only. You should then save this to tape or disc.

Next, install the program to be debugged into the Text area. You will need to check that the program has no lines greater in length than a full screen line, and that the first program line starts at the very start of the Text area. While you are experimenting with the debugger, it will be as well to install a fairly simple program for starters.



Now press Shift-f6 to run the debugger. You should see a menu appear as in the illustration. Option D allows you to view (or print out) the program to be debugged with line numbers added for reference.

Option E will simply Exit to the Text area. It is option C which does all the work. If you select this you will hear a series of beeps, one for each of your program lines, as the debugger copies them into Segment 7, after first clearing that area. In between each line of your copied code it inserts two other lines, and appends a block of code at the end. this produces a 'compiled' version of your segment program, with extra code added to allow single-stepping through the program.

Once the compilation is complete, you will be prompted to begin single-stepping. If you answer with "Y", then the compiled program in Segment 7 will be automatically run. This will initiate single-stepping, and you will see the first line of your program printed in a window at the base of the screen, along with the values of selected variables (A%, B% and A$ initially). Further presses of the space bar will have the effect described before.

Once the last line of your program has been executed, you will be left in the currently selected segment. To re-run the debugger, you simply press Shift-f6. If you do this you will see that the debugger always makes a fresh compilation before commencing to single-step. This avoids the problems that would arise if the compiled version of your program differed from that held in the Text area. But if you have made no changes, you can restart single-stepping with Shift-f7.

SELECTED VARIABLES

You will notice that the selected variables displayed by the debugger during a single-stepping run are A%, B% and A$. To alter these, you should change the occurrences of A%, B% or A$ on lines 81, 82 or 83, as appropriate.

RESTRICTIONS

There are very few restrictions on the program to be debugged in this way, but it must clearly NOT use the same segments (6, 7 and the Text area) or the same variables as the compiled debugging program, otherwise confusion will result. Fortunately the number of variables used by the debugger itself has been kept to just four. They are X%, Y%, Z% and Z$. If your program uses any of these, you should either globally alter them in your own program or in the debugger. In the latter case, this will ensure that they do not appear in the compiled debugging program. You do not need to avoid the additional variables used in the compiler itself (such as B$, C% etc). These do not appear in the compiled program, and will therefore cause no problems at run time.

HINTS

If your program uses the GET function, single-stepping using the space-bar will always return character 32 (space). But the debugger allows you to use any key to single-step. Therefore, simply use the key corresponding to the required response to the GET that you are trying to trace. Thus if you use the debugger on itself, pressing "E" instead of the space bar will eventually take you to the EXIT option on the menu. You cannot of course debug the Compile option because of the inherent clash of segments.

```
1    REM WORDWISE DEBUG
2    REM VERSION Ø.3 DEG
3    .START
4    VDU7
5    B$=CHR$(13)
6    CLS
7    VDU 31,4,6,134,141
8    P."WORDWISE PLUS DEBUG"
9    VDU31,4,7,134,141
10   P."WORDWISE PLUS DEBUG"
11   P.
12   P.
13   P.
14   VDU131
15   P."C   COMPILE SINGLE STEPPER"
16   P.
17   VDU131
18   P."D   DISPLAY SOURCE PROGRAM"
19   P.
20   VDU131
21   P."E   EXIT TO SOURCE PROGRAM"
22   P.
23   *FX15
24   C$=GCK$
25   IF C$="C" THEN GOTO COMP
26   IF C$="D" THEN PROC DISP
27   IF C$="E" THEN GOTO EXIT
28   GOTO START
29
30   .COMP
31   X%=1
32   M%=2
33   SELECT SEGMENT 7
34   CURSOR TOP
35   FKEY 3
36   CURSOR BOTTOM
37   FKEY 3
38   DELETE MARKED
39   CURSOR TOP
40   SELECT TEXT
41   CURSOR TOP
42   REPEAT
43   Z$=GLT$
44   SELECT SEGMENT 7
45   CURSOR TOP
46   CURSOR DOWN M%
47   TYPE "X%="+STR$(X%)+B$
48   TYPE "PROCninst"+B$
49   TYPE Z$+B$
50   SELECT TEXT
51   CURSOR TOP
```

```
52   CURSOR DOWN X%
53   X%=X%+1
54   M%=M%+3
55   UNTIL EOT
56   REM ADD PROCNinst TO TRACER
57   SELECT SEGMENT 7
58   TYPE ".Ninst"+B$
59   TYPE "IF X%>1 THEN Z%=GET"+B$
60   TYPE "VDU7"+B$
61   TYPE "IF X%>1 THEN DISPLAY"+B$
62   TYPE "IF X%>1 THEN Z%=GET"+B$
63   TYPE "VDU7"+B$
64   TYPE "Y%=?&67-&30"+B$
65   TYPE "SELECT TEXT"+B$
66   TYPE "CURSOR TOP"+B$
67   TYPE "DOTHIS"+B$
68   TYPE "CURSOR DOWN"+B$
69   TYPE "TIMES X%"+B$
70   TYPE "CURSOR UP"+B$
71   TYPE "Z$=GLT$"+B$
72   TYPE "CLS"+B$
73   TYPE "VDU28,0,24,39,16"+B$
74   TYPE "DOTHIS"+B$
75   TYPE "VDU129,157,134,13"+B$
76   TYPE "P."+B$
77   TYPE "TIMES 7"+B$
78   TYPE "VDU28,3,24,39,16"+B$
79   TYPE "P.STR$(X%)+""  ""+Z$"+B$
80   TYPE "P."+B$
81   TYPE "P.""A% = ""+STR$(A%)"+B$
82   TYPE "P.""B% = ""+STR$(B%)"+B$
83   TYPE "P.""A$ = ""+A$"+B$
84   TYPE "VDU26"+B$
85   TYPE "Z%=GET"+B$
86   TYPE "VDU7"+B$
87   TYPE "SELECT SEGMENT Y%"+B$
88   TYPE "ENDPROC"+B$
89   VDU7,7,7
90   P.
91   P.
92   VDU134
93   P."CODE IS COMPILED"
94   VDU134
95   P."BEGIN SINGLE STEP ? (Y/N)"
96   Z%=GET
97   IF Z%<>89 AND Z%<>121 THEN GOTO START
98   *FX15
99   O.("FX138,0,151")
100  END
101
102  .DISP
103  H%=FALSE
104  PRINT
105  VDU131
106  PRINT "HARD COPY? (Y/N) ";
107  .ENTER
108  C$=GCK$
109  IF C$="Y" THEN H%=TRUE
110  IF C$="N" THEN VDU 14
111  IF C$<>"N" AND C$<>"Y" THEN GOTO ENTER
112  SELECT TEXT
113  X%=1
114  CURSOR TOP
115  CLS
116  IF H% THEN VDU 2
117  REPEAT
118  N$=STR$(X%)
119  P. N$;
120  DOTHIS
121  P." ";
122  TIMES 5-LEN N$
123  P. GLT$
124  CURSOR AT 0
125  X%=X%+1
126  UNTIL EOT
127  VDU 3,15
128  P.
129  P."Press any key"
130  C$=GCK$
131  ENDPROC
132
133  .EXIT
134  VDU7
135  SELECT TEXT
136  DISPLAY
137  END
```

POINTS ARISING POINTS ARISING POINTS ARISING POINTS

BEEBUG FILER (BEEBUG Vol.4 Nos.6 to 8, Vol.5 No.1)

A small bug has emerged in the sort routine. Change lines 8980 and 9000 as follows:

```
8980 R=0:REPEAT:R=R+1:UNTIL p$=field$(R) OR R=f
9000 IF R=f AND p$<>field$(R) PRINT"No such field":ENDPROC
```

A similar bug also exists in the FNfield function which can be corrected by changing the following two lines:

```
21620 REPEAT:I=I+1:UNTIL p$=field$(I) OR I=f
21640 IF I=f AND p$<>field$(I) THEN =" " ELSE =record$(I,R)
```

The mail-merge feature described last month also led to a small bug being introduced into the display of records on the screen. Change line 4980 as shown:

```
4980 IF I<end AND flag=0 THEN G=GET
```

MASTER SERIES

Opening up the Private RAM

Many Beeb owners are upgrading to Acorn's new Master series. We start our own Master series by taking a look at private RAM with Thomas Nunns.

Programming the ten function keys can remove much of the routine typing needed when a program is being written or used. The function keys can include programming instructions - RENUMBER, MODE 135, LIST - filing system commands - *COMPACT or *FORM 80 0 2 - or software instructions such as printer codes for use with a wordprocessor.

With both the Master and the older Model B/B+ the *KEY command can be used to program the function keys in Basic. On the older BBC micro the unexpanded function key definitions are held in main memory between &B00 and &BFF, so to save them as a separate file it is only necessary to type *SAVE <filename> B00 BFF, and they can then be reloaded, without re-programming, using *LOAD <filename>.

On the MASTER things are not so simple. Function key definitions are no longer held in main memory but in the so-called 'Private RAM' which sits between &8000 and &DFFF. The allocation of memory in this sideways RAM area is shown in the table below.

MASTER 128 Private RAM Usage

&8000-&83FF	Function key buffer
&8400-&87FF	VDU workspace
&8800-&88FF	VDU variables and workspace
&8900-&8FFF	Character definitions
&C000-&DCFF	Paged ROM workspace
&DD00-&DFFF	MOS workspace

Acorn say that this 'Private RAM' should not be used directly, and so they have not provided any read and write calls to access the area directly. This makes the direct saving and loading of the function key definitions awkward. It would be possible to save them by using the *SHOW command and spooling the results to a file:

```
*SPOOL keys
FOR C%=0 TO 15
OSCLI("SHOW "+STR$(C%))
NEXT
*SPOOL
```

But to reload them the file would have to be read in a character at a time and a check made for the beginning and end quotes of each definition before using a OSCLI("KEY "+ .....) command.

However, there is another 'illegal' way. The Operating System sees the Private RAM as a ROM when bit 7 of the ROM select flag at &F4 is set. By setting the ROM select flag and its copy in SHEILA to 128, the Private RAM can be written to. In addition, by using the (undocumented by Acorn) Operating System Read ROM routine (OSRDRM), the Private RAM can be read as ROM number 128.

SAVING THE DEFINITIONS

Program 1 saves the entire function key buffer in the Private RAM (from &8000 to &83FF) to a file. By saving the whole area the flags held at the beginning of the function key buffer are also saved. The Private RAM is accessed by setting the current ROM flag to 128 (line 210). The routine then reads the bytes from the Private RAM into main memory (&7000 to &7400) as if it were a sideways ROM, using the Operating System Read ROM (OSRDRM) routine at &FFB9 (see later about use of main memory). Once the bytes are in main memory they are saved to a file (line 60).

PROGRAM 1 - SAVING THE KEYS

```
 10 MODE7
 20 INPUT"Filename for definitions",f$
 30 DIM mc 100
 40 PROCass
 50 CALL mc
 60 OSCLI("SAVE "+f$+" 7000 7400")
 70 END
 80 DEFPROCass
 90 FOR o%=0 TO 2 STEP2
100 P%=mc
110 [OPT o%
120   STZ &80      \ set store at &7000
130   LDA #&70
140   STA &81
150   LDA #&80     \ high byte for OSRDRM
160   STA &F7
170 .four
```

```
180   LDY #0
190 .loop
200   STY &F6      \ low byte for OSRDRM
call
210   LDY #128     \ ROM number 128
220   JSR &FFB9    \ call OSReaDRoM
230   LDY &F6      \ get result
240   STA (&80),Y \ store byte in main
250   INY          \ memory for four
260   CPY #0       \ pages
270   BNE loop
280   INC &81
290   INC &F7
300   LDA &F7
310   CMP #&84
320   BNE four
330   RTS
340 ]
350 NEXT
360 ENDPROC
```

RE-LOADING THE DEFINITIONS

Program 2 restores the function key definitions by loading them into main memory and then transferring them to the Private RAM. Whereas it would be quite easy to save the keys a byte at a time without using main memory, this is not easy when reloading the definitions as the Private RAM occupies the currently selected ROM slot. To put back the keys a byte at a time, the Private RAM and the DFS ROM would have to be paged in and out alternately. This might sound easy but there are a number of bytes which would have to be saved and replaced in the MOS workspace and in SHEILA to make it work. Instead &400 bytes of main memory are used as a buffer, and the final lines of the code (360-440) restore the Basic pointer and recall the Basic ROM. This use of main memory may or may not be acceptable in some cases. Once back in Basic, any other language, such as View, can be called without losing the definitions.

PROGRAM 2 - LOADING THE KEYS

```
 10 MODE7
 20 INPUT"Filename of definitions",f$
 30 DIM mc 100
 40 PROCass
 50 OSCLI("LOAD "+f$+" 7000")
 60 CALL mc
 70 END
 80 DEFPROCass
 90 FOR o%=0 TO 2 STEP2
100 P%=mc
110 [OPT o%
120   STZ &80      \ &80 and &82 are
130   STZ &82      \ pointers for PRIVATE
140   LDA #&70     \ RAM and main memory
150   STA &81      \ respectively.
160   LDA #&80
170   STA &83
180   LDA #128     \ set active ROM
190   STA &F4      \ flag and its copy
200   STA &FE30    \ in SHEILA to 128.
210 .four
220   LDY #0
230 .loop
240   LDA (&80),Y  \ move from Private
250   STA (&82),Y  \ RAM to main memory
260   INY          \ a byte at a time.
270   CPY #0
280   BNE loop
290   INC &81
300   INC &83
310   LDA &81
320   CMP #&74
330   BNE four
340   LDA #&97     \ reset ROM latch
350   LDX #30      \ in SHEILA to
360   LDY #&0C     \ Basic ROM.
370   JSR &FFF4
380   LDA #&0C     \ reset latch at &F4
390   STA &F4      \ to Basic ROM.
400   LDA #&8E     \ do *BASIC
410   LDX #&0C
420   JSR &FFF4
430   RTS
440 ]
450 NEXT
460 ENDPROC
```

As the programs stand, they must be loaded and run. This is both for simplicity and so that their workings may be more easily understood. Those with the wisdom and the will could easily alter them so that the code is assembled at &7500 and *SAVED. This would allow the key definitions to be saved and loaded with a *RUN <filename> while another Basic program remains uncorrupted in memory below &7000. Even so, after using the re-load code an OLD will have to be issued to reset the Basic pointers. To do this the few Basic lines at the start of each program would also have to be recoded in assembler. For the really clever, the re-load code can be *SAVEd along with the definitions and a suitable execution address. This enables the keys to be reloaded with a single disc access!

If you have a Master then you may be able to contribute to this series. Please write to the editor with details of any relevant programs or articles.

PASSING ARRAYS TO PROCEDURES

Although BBC Basic is one of the best, its procedures are still quite limited when it comes to passing parameters. John Skingley adds the power you need.

It is an often felt limitation of Basic, that the BEEB's procedure and function calls do not permit arrays to be passed as parameters, nor do they allow parameter values to be passed back to the calling routine. Functions do of course return values, but even these can only return one value at a time. Both of these limitations may be overcome by the methods described here.

The two main methods of passing parameters to procedures are often referred to as "by value", and "by reference". BBC Basic uses the first method only.

Passing parameters by value involves copying the contents of the variable held in the global variable space, into the local data space for the procedure, where it is treated just like any other local variable. If the variable is changed in value, only the copy stored in this local data space is changed, so that when the procedure terminates, the changed value is lost (i.e. the entry value is retained).

To pass an array to a procedure by this method, the whole array would have to be copied into the local data space. This would take a lot of time and memory space.

In the second method, passing by reference, only a pointer to the variable, stored in the global variable space, is passed to the procedure. In this way the procedure may 'refer' to this global variable and not only read, but also change its value. However, (unlike passing by "value" where if the value of a parameter was changed within the procedure it was unavailable outside) the global variable used as a parameter is actually changed - its value is available for use by the calling routine after the procedure has finished with it. This is the method we would need to use if we wanted, for example, to pass an array to a procedure which sorted the array into order, or carried out some other operation on it.

BBC Basic has no inherent operation to specify that we require parameters to be passed by reference, unlike many other high level languages. However, the BEEB has a simple facility for storing data in memory, and accessing this by the use of a pointer. This is set up with the "DIM p 4" statement, which allocates, in this case, 4 bytes of memory for the data, where p holds the address of the starting memory location used. In other words p is the pointer to the data. Values may be entered into this location by the use of the indirection operators, with statements such as ?p=34, and retrieved with, for instance, PRINT ?p. The ? indirection operator only deals with single bytes. The ! indirection operator is designed to handle 4 bytes and thus is ideal for handling integers. In BEEBUG Vol.3 Nos. 7 & 8 the topic of indirection operators was covered in detail, I would strongly suggest that you read these after having read this article if you are in any way uncertain of their use.

Using this method of storing variables, we can pass the pointers as parameters in procedure calls, carry out the required action on the data pointed to, and still access the changed values afterwards. We can even create arrays of variables, (or more complicated data structures), and pass their pointers to our procedures. This may sound complicated, but an examination of the two programs in the example that follows, should indicate how straightforward this is in reality.

EXAMPLE ONE

To illustrate the use of such variables, and to show how they can be passed to procedures look at the two near-identical programs:

PROGRAM ONE:

```
  20 @%=6 : b=0 : c=0
  30 FOR a=1 TO 10
  40   PROCcalculate(a,b,c)
  50   PRINT a,b,c
  60 NEXT : END
 100 DEFPROCcalculate(a,b,c)
 110  b=a*a : c=a*b : ENDPROC
```

PROGRAM TWO:

```
10 DIM a 4,b 4,c 4
20 @%=6 : !b=0 : !c=0
30 FOR !a=1 TO 10
40   PROCcalculate(a,b,c)
50   PRINT !a,!b,!c
60 NEXT : END
100 DEFPROCcalculate(a,b,c)
110  !b=!a*!a : !c=!a*!b : ENDPROC
```

Both programs are intended to produce a table of numbers, squares, cubes and square roots. However, only the second one produces the desired results. Notice how we can easily substitute !a for a, !b for b, etc, while we can of course use meaningful names such as !square !cube etc. There is nothing frightening about the ! and ? operators once you have gained familiarity with them. Note that the program would have worked perfectly with the ? operator, until the numbers stored exceeded 255 as the ? operator handles only a single byte.

ARRAYS

A single dimensional array can be thought of simply as a list. If we wish to hold a list of integers we must reserve four bytes for each one. This is handled automatically if you use DIM A%(10) - using this method you do not need to do any further work if you wish to fill the array with random numbers and print them. The program on the left does just that.

```
10 DIM A%(10)          10 DIM A 44
20 FOR i=0 TO 10       20 FOR i=0 TO 10
30 A%(i)=RND(1000)     30 A!(i*4)=RND(1000)
40 PRINT A%(i):NEXT    40 PRINT A!(i*4):NEXT
```

To produce an identical array using the pointer system you must first allow each of the 11 integers (0-10) the four bytes they require, i.e. 44 bytes and declare this as DIM A 44. To access the elements of the array you must note that:

```
Element 0 is held in 4 bytes starting at 0
        1                                 4
        2                                 8
        i                               i*4
```

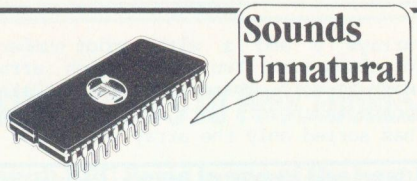
The program on the right performs an identical task to that on the left. Note that the format A!(i*4) is identical to !(A+i*4) in other words read four bytes starting at address (A+i*4).

CONCLUDING EXAMPLE

In this example the program fills two arrays y and z with random numbers and displays each. You select which array you wish to sort. The program sorts that array only and prints both arrays to prove it has sorted only the array requested.

```
10 Reserve adequate memory for two arrays
   (in fact DIM y 40 will do if n=10) and
   also for two working variables b and c.
30 Set max no of elements and PRINT format
60 Print headings
70 Fill arrays with random numbers
80 Print the filled arrays
90-120 Input which array is to be sorted
130 Sort selected array only
140 Print both arrays
PROCEDURES:
1000 Fill array with random integers
2000 Print array in specified row & column
3000 Bubble sort
4000 Swapping contents of two variables
```

```
10 DIM y 100, z 100, b 4, c 4
30 n=10:@%=6
60 CLS:PRINT"ARRAY Y"TAB(20)"ARRAY Z"
70 PROCgenerate(y):PROCgenerate(z)
80 PROCprint(y,0,1) :PROCprint(z,20,1)
90 PRINT TAB(0,11)"Sort which array 'y' o
r 'z' ? "TAB(30,11);
100 arr$=CHR$(GET AND 223)
110 IF arr$<>"Y" AND arr$<>"Z" THEN 90
120 PRINT arr$
130 IF arr$="Y" PROCsrt(y) ELSE PROCsrt(z)
140 PROCprint(y,0,12):PROCprint(z,20,12)
990 END
995 :
1000 DEFPROCgenerate(a)
1010 FOR i=1 TO n
1020   a!(i*4)=RND(100000)
1040 NEXT i : ENDPROC
1060 :
2000 DEFPROCprint(a,column,row)
2010 FOR i=1 TO n
2020   PRINT TAB(column,i+row-1)a!(i*4)
2030 NEXT i : ENDPROC
2050 :
3000 DEFPROCsrt(a)
3010 swapped=FALSE : m=n-1
3020 REPEAT
3030   FOR i=1 TO m
3040     IF a!(i*4)>a!((i+1)*4) PROCswap(
a,i*4,(i+1)*4)
3050   NEXT i
3051   m=m-1
3060 UNTIL NOT swapped OR m=1
3070 ENDPROC
3080 :
4000 DEFPROCswap(a,b,c)
4010   t=a!b:a!b=a!c:a!c=t
4015   swapped=TRUE : ENDPROC
```



Never at a loss for words, Alan Webster has been testing out the latest speech add-on from Computer Concepts, with a sideways look at Superior Software's own speech system.

Product : SPEECH!
Supplier : Superior Software
Regent Ho., Skinner Lane,
Leeds 7. (0532) 459453.
Price : £9.95 (tape)
£11.95 (disc)

Product : Text to Speech
Supplier : Computer Concepts
Gaddesden Place, Hemel
Hempstead, Herts., HP2 6EX.
(0442) 63933
Price : £39.90 excl. speech processor
£10.00 speech processor

One of the major advantages that the Computer Concepts' Speech package had over many others on the market was that it used the TMS5220 speech chip, giving much clearer speech. It was not one of the easiest of systems to use, however, as text had to be typed in using phonemes (see review in BEEBUG Vol.4 No.6). Although this was adequate for most purposes, it made it difficult to encode phrases. Now Computer Concepts have added the 'Text to Speech' (TTS) ROM to the original speech package to provide the 'SPEECH SYSTEM'.

The new package contains both the Speech ROM and the new TTS ROM, but as before, it is essential that the TMS5520 speech chip is fitted as well. If you already have the original Computer Concepts Speech ROM, an upgrade service is available (from Computer Concepts) for £11.75, but this then brings the total cost of the system to over £55.

The main feature of TTS is a *SAY command which will pronounce any text string that follows. This is much easier to use than building up a set of words by using phonemes. For example, '*SAY Hello there' is far easier than the '*UTTER <1> H +e L O DH *ea' command needed previously. You can also set the TTS into one of three different modes, to pronounce words, spell them out, or pronounce lower case words and spell out upper case words. The TTS ROM itself takes 1K of memory as workspace.

Other commands included are *PHONS, which acts the same as *SAY, but lists the phonemes needed to construct the speech. *SAYIP allows the TTS to pronounce any text entering the current input stream, such as the keyboard or RS423 interface, both of which offer some interesting possibilities.

*SAYPRT diverts the text from a printer and pronounces it. In this mode, TTS can be switched on and off using VDU 2 and VDU 3. There are also commands to pronounce the text on screen (similar to a text printer dump) and to pronounce a section of memory or file.

A nice feature of the TTS is that various standard abbreviations are pronounced fully, 'Mr. Williams' sounding as 'Mister Williams' for example.

Superior Software's Speech! adopts a completely different approach to speech generation and in this respect is a much more innovatory piece of software. In fact, it does not use the Beeb's speech system at all. The whole thing is handled in software and a stream of data is pushed out through the Beeb's sound generator. As a consequence, this package is considerably cheaper, costing only £9.95 as opposed to nearly £50.

In spite of the very cunning and unorthodox route to speech generation taken by Superior, the results are really quite good, though the clarity does not quite match up to Computer Concepts' chip-based speech. Speech! also uses a *SAY command for ease of use, but does not offer as many star commands as Computer Concepts.

If you are looking for a speech system simply for home entertainment then with its dramatically lower price Speech! must be the better bet. If you seek a serious speech system, where speech quality is paramount, then the Computer Concepts product is the one to go for.

Computer Concepts Roms
NEW BEEBUG PRICES

Inter Sheet — The Spread Sheet
RRP £52.35 ..... **NOW £47.70**

Inter Chart — Graphs & Plotting
RRP £36.80 ..... **NOW £29.90**

Disc Doctor — The Disc Rom
RRP £33.35 ..... **NOW £28.00**

Printmaster — The Printer Rom
(Specify Epson or Star)
RRP £33.35 ..... **NOW £28.00**

Wordwise —
Original Wordprocessor
RRP £46.00 ..... **NOW £29.00**

Wordwise Plus —
The New Wordprocessor
RRP £56.35 ..... **NOW £46.00**

Hi-Wordwise Plus — Disc ... **£5.75**
(Requires Wordwise Plus)

Hi-Intersheet — Disc **£5.75**
(Requires Intersheet)

Accelerator — 24K Compiler
RRP £64.40 ..... **NOW £56.00**

Graphics Rom —
BBC Extended Graphics
RRP £33.95 ..... **NOW £25.00**

Speech Rom — Speech System
RRP £33.35 ..... **NOW £28.00**
(Requires Acorn speech chip available at extra £10.00 only if ordered with Speech Rom)

All Prices Include VAT and Post.

For a full Specification of all Roms send an A5 SAE marked Rom Spec.

Members, please note that we will match any currently advertised price in Acorn or Micro User for these products from recognised dealers.

All orders despatched within 24 hours of receipt at St. Albans.

Buy more than 1 Rom and get £1.00 off each.

COMPUTER CONCEPTS ROM ORDER FORM

Please send me _____ price _____

_____ _____

_____ _____

BEEBUG SOFT

TOTAL ENCLOSED _____

Credit Card orders may be telephoned to (0727) 40303

Prices include post, packing and VAT, overseas members send the same amount to include extra post but not VAT. Please make cheques payable to BEEBUGSOFT.

***Send to:* "Rom Offer" Beebugsoft, PO Box 50, Holywell Hill, St. Albans, Herts. AL1 3YS**

To avoid delay please do not send this to the High Wycombe address, please do not combine orders for other BEEBUGSOFT products with this order.

*Name*_____ *Membership No.*_____

*Address*_____

_____

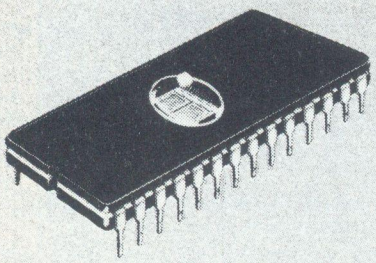
Access/Visa Card No. | | | | | | | | | | | | | | | | *Expiry date*

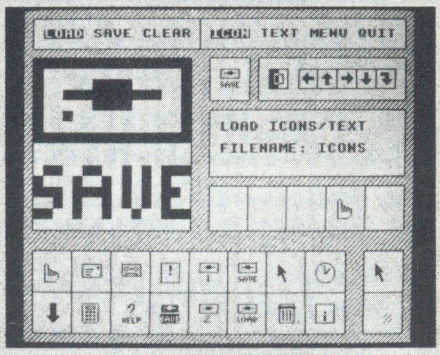
ICON MASTER FROM BEEBUGSOFT

Icons are graphical representations of functions. More powerful mini and micro computers use icons in place of the more traditional menus as a front end for many of their programs.

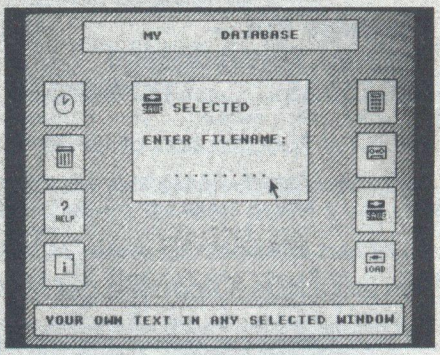
For example, an icon of a dustbin could be used to represent the *DELETE command, or a clock to ask for the time. Icon Master enables you to add the power of icons to your own programs. Replace your menu screens with dynamic screens of icons, where selection is controlled from the cursor keys. Icons are sophisticated but simple to use, as anyone who has used an AMX Mouse will know.

Icon Master is normally controlled from the keyboard, but AMX Mouse owners may prefer to control it with their AMX Mouse, Icon Master automatically knows which you are using. Once you have created your icon front end, you may use it with or without Icon Master in your computer and so may pass it to friends or even include it in software that you intend to sell.





Icon Master is easy to use. Firstly, you create your own Icons on the (icon controlled) designer screen.



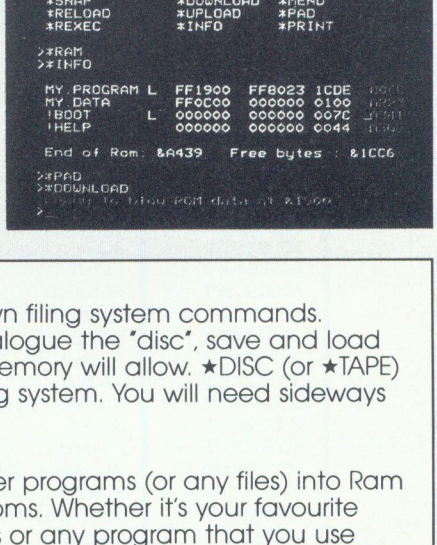
Then you create your own screen and position your icons anywhere on it. Once you have finished, Icon Master creates the necessary code and adds it to your program for you. This screen shows how you might use icons in a database.

BEEBUG SOFT

16K Rom **£34.00** Members **£25.50**

ROMIT — Put your own basic programs into Rom

- **Create Your Own Silicon Disc**
- **A Complete Filing System with up to 16K capacity (per bank of Ram)**



ROMIT adds a new feature to your BBC Micro — A Ram Filing System. This provides over 20 new commands which enable you to:

1. Create a silicon disc with its own filing system commands. Type ★RAM, then you can catalogue the "disc", save and load files to it, storing as many files as memory will allow. ★DISC (or ★TAPE) takes you back to your normal filing system. You will need sideways Ram to use this facility.

2. Put your own Basic or assembler programs (or any files) into Ram and then make them into Eproms. Whether it's your favourite game, your function key definitions or any program that you use frequently, you can now put it into Eprom so that it's always instantly available. Multiple copies can also be produced to enable you to pass on your programs to other people.

Getting your own Basic program into Rom couldn't be simpler. Just load your program into memory and type:

★RAM ★TRANSFER "prog" ★DOWNLOAD

and either connect it to a blower, or send the disc (or tape) to us, together with your **RomIt** registration number, and we will Rom it for you. (There is a small charge for this service).

3. Use your sideways Ram as a printer buffer so that you can continue to use your computer for other work at the same time as printing long documents.

OVER 20 NEW ★COMMANDS ARE PROVIDED BY ROMIT

MEMBERS PRICE £25.50

An Eprom Blowing Service is offered with this pack.

To get the most from **RomIt** you will need sideways Ram fitted to your computer, eg just a single Ram chip plugged into your ATPL (or other) Rom board.

NEW

High Scores

Supplier	Game	Score	Player
AARDVARK	Zalaga	15007810	P.O'Malley
AARDVARK	Frak!	20000200	P.O'Malley
A&F	Chuckie Egg	25000390	P.O'Malley
A&F	Cylon Attack	77730	P.O'Malley
A&F	Painter	160600	M.Shaw
A&F	Planes	57485	A.Butcher
A&F	Plank Walk	7970	D.Hesketh
ACORNSOFT	Arcadians	40220	P.Schmedling
ACORNSOFT	Aviator	9680	C.Bunch
ACORNSOFT	Boxer	45640	J.Richardson
ACORNSOFT	Carousel	44670	E.Somerville
ACORNSOFT	Freefall	17745	I.Teixeira
ACORNSOFT	Hopper	62100	R.Barnes
ACORNSOFT	Invaders	26260	P.Orton
ACORNSOFT	JCB	23690	C.Bunch
ACORNSOFT	Labyrinth	510300	Cthulhu
ACORNSOFT	Meteors	58290	D.Wigg
ACORNSOFT	Missile Base	173590	M.Aggarwal
ACORNSOFT	Monsters	257060	I.Cook
ACORNSOFT	Planetoid	875185	J.De Saram
ACORNSOFT	Revs	1:24.6	D.Whittam
ACORNSOFT	Rocket Raid	200050	V.Nicholas
ACORNSOFT	Snake	1410	S.Werrett
ACORNSOFT	Snapper	1089270	S.Murphy
ACORNSOFT	Starship Command	11112	A.Bailey
ACORNSOFT	Super Invaders	68420	G.Sands
ACORNSOFT	Tetrapod	8450	I.Melville
ACORNSOFT	Tracer	367000	P.O'Malley
ACORNSOFT	Volcano	6150	R.F.Tagart
ADDICTIVE	Boffin	227570	J.Longhurst
ADDICTIVE	Boffin 2	308115	E.Longhurst
ALLIGATA	Blagger	18830	K.Butler
ALLIGATA	Bug Blaster	539348	K.Allen
ALLIGATA	Eagle Empire	99300	A.Hayden
ALLIGATA	Son of Blagger	79960	K.Butler
ASP Software	Froglet	175770	G.Deas
ATARISOFT	Crystal Castles	26700	J.Tompkins
ATARISOFT	Donkey Kong Jnr.	19400	J.Tompkins
ATARISOFT	Joust	1410565	S.Mughal
ATARISOFT	Pole Position	113100	J.De Saram
ATARISOFT	Robotron 2084	167500	S.Menges
BEEBUG GAMES	Santa	43420	S.Williams
BEEBUG MAG	Aliens	890	R.Phillips
BEEBUG MAG	Block Blitz	34510	D.Tilley
BEEBUG MAG	Breakout	21950	L.Chasmer
BEEBUG MAG	Brickie Nickie	97330	A.Hockton
BEEBUG MAG	Cosmonaut	4040	R.Koten
BEEBUG MAG	Detonator Dan	1119515	R.Tweed
BEEBUG MAG	Dive Bomber	42280	J.Hanrahan
BEEBUG MAG	Elevasion	25600	D.Tilley
BEEBUG MAG	Galactic Invasion	96300	A.Hockton
BEEBUG MAG	Hedgehog	32760	C.Eberhardt
BEEBUG MAG	Invasion	750	D.Tilley
BEEBUG MAG	Manhole	224	G.Deas
BEEBUG MAG	Munchman	62596	D.Duffey

→6

LOW COST C.A.D.

ATTENTION ALL ELECTRONICS CIRCUIT DESIGNERS!!
IBM PC, BBC MODELS B and SPECTRUM 48K

ANALYSER I and II compute the A.C FREQUENCY RESPONSE of linear (analogue) circuits. GAIN and PHASE, INPUT IMPEDANCE, OUTPUT IMPEDANCE, and GROUP DELAY (except Spectrum version) are calculated over any frequency range required. The prgrams are in use regularly for frequencies between 0.1Hz to 1.2GHz. The effects on performance of MODIFICATIONS to both circuit and component values can be speedily evaluated.

Circuits containing any combination of RESISTORS, CAPACITORS, INDUCTORS, TRANSFORMERS, BIPOLAR AND FIELD EFFECT TRANSISTORS and OPERATIONAL AMPLIFIERS can be simulated — up to 60 nodes and 180 components (IBM version).

Ideal for the analysis of ACTIVE and PASSIVE FILTER CIRCUITS, AUDIO-AMPLIFIERS, LOUDSPEAKER CROSS-OVER NETWORKS, WIDE-BAND AMPLIFIERS, TUNED R.F. AMPLIFIERS, AERIAL MATCHING NETWORKS, TV I.F. and CHROMA FILTER CIRCUITS, LINEAR INTEGRATED CIRCUITS etc.

STABILITY CRITERIA AND OSCILLATOR CIRCUITS can be evaluated by "breaking the loop".

Tabular output on Analyser I. Full graphical output, increased circuit size and active component library facilities on Analyser II.

Check out your new designs in minutes rather than days.

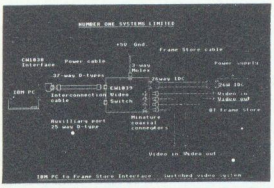
ANALYSER can greatly reduce or even eliminate the need to breadbord new designs.

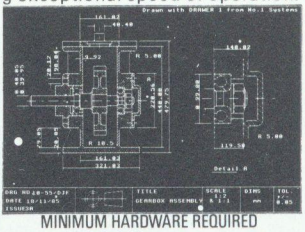
Full AFTER SALES SERVICE with TELEPHONE QUERY HOT LINE and FREE update service.

Used by INDUSTRIAL, GOVERNMENT, and UNIVERSITY R & D DEPARTMENTS worldwide. IDEAL FOR TRAINING COURSES. VERY EASY TO USE. Prices from £20 — £195.

DRAUGHTING
BBC MODEL B

"DRAWER I" enables quality drawings to be created, and modified, quickly, easily and with the minimum of hardware. Positional input is by standard joystick. All of the major program elements are written in machine code giving exceptional speed of operation.





FEATURES

- ☆ Rubber Banding for drawing lines:
- ☆ Solid or Dotted lines types.
- ☆ Circles, arcs and partial or complete elipses.
- ☆ Vertical or Horizontal Text.
- ☆ Pan and Zoom.
- ☆ Merging of drawings and library symbols from disc.
- ☆ Snap to a user defined grid.
- ☆ Absolute or Relative cursor co-ordinates displayed on screen.
- ☆ Output to standard dot matrix printer.
- ☆ Price — From £45 ex. VAT.

MINIMUM HARDWARE REQUIRED

- ☆ BBC Model B.
- ☆ Single or Dual Disc Drive — 40 or 80 track.
- ☆ T.V. or monitor.
- ☆ Games Joystick with "fire button".
- ☆ Dot Matrix Printer (Epson 80 series or Epson compatible — BBC default mode preferable).

For illustrated leaflets and ordering information please contact:
NUMBER ONE SYSTEMS LIMITED
Ref: B.B.
Crown Street,
St. Ives Huntingdon, Cambs PE17 4EB

TEL: 0480 61778
TELEX: 32339

←4

BEEBUG MAG	Robots	31510	D.Tilley
BEEBUG MAG	Scrumpy	39975	A.Webster
BEEBUG MAG	Space City	110362	R.Bunnett
BEEBUG MAG	Truffle Hunt	6189	R.Koten
BEEBUG MAG	Wee Shuggy	5124	G.Togwell
BEEBUGSOFT	Astro-tracker	38555	N.Hallam
BEEBUGSOFT	Magic Eel	8000	A.Webster
BEEBUGSOFT	Snorter	33202	G.Deas
BEEBUGSOFT	Swarmers Revenge	92800	A.Webster
BUG-BYTE	Oblivion	5010	Cthulhu
BUG-BYTE	Sealord	4020	M.Orton
COMP CONCEPTS	Android Attack	1132985	J.Button
DR.SOFT	Pengo	98000	C.Chamberlain
DURRELL	Combat Lynx	1290	J.Morley
DYNABYTE	Space Ranger	25450	P.Orton
DYNABYTE	Demolator	12600	P.Orton
FIREBIRD	Acid Drops	19940	I.Melville
FIREBIRD	Bird Strike	37910	I.Melville
FIREBIRD	Duck	55200	E.Denning
FIREBIRD	Gold Digger	25127	E.Somerville
FIREBIRD	Hacker	7138	P.Orton
GEMINI	Caterpillar	52880	B.E.Atthat
GEMINI	Missile Control	97450	F.Fook
H SOFT	Penguin	163730	J.Dowling
HOPESOFT	Escape from Orion	100140	P.Orton
ICON	Caveman Capers	23800	I.Melville
ICON	Contraption	4720	I.Melville
ICON	Drain Mania	80400	Cthulhu
IMAGINE	Yie-Ar Kung Fu	420000	J.Jackson
IMAGINE	Hypersports	147260	A.Webster
KANSAS	Galactic Firebird	46850	D.Wigg
MICRO-BYTE	3D Space Ranger	334500	R.Bunnett
MICRO POWER	Asteroid Storm	44410	J.Terry
MICRO POWER	Bumble Bee	31900	R.White
MICRO POWER	Castle Quest	2210	S.Rawnsley
MICRO POWER	Croaker	42830	S.Menges
MICRO POWER	Cybertron Mission	164500	P.Luqman
MICRO POWER	Danger! UXB	1100700	D.Russell
MICRO POWER	Demon Decorator	92500	J.Drysdale
MICRO POWER	Dune Rider	8790	J.Button
MICRO POWER	Frenzy	1341961	J.Sims
MICRO POWER	Killer Gorilla	472700	S.Chasmer
MICRO POWER	Felix in the Factory	23280	G.Stubbs
MICRO POWER	Felix and Monsters	95540	Joe
MICRO POWER	Ghouls	6100	D.Garvie
MICRO POWER	Jet Power Jack	18275	Cthulhu
MICRO POWER	Laser Command	1010525	C.Eberhardt
MICRO POWER	Martian Attack	32300	Cthulhu
MICRO POWER	Mine	224640	Cthulhu
MICRO POWER	Moonraider	232200	C.Shilling
MICRO POWER	Mr.Ee	480600	Mrs.Crystal
MICRO POWER	Positron	276850	A.Hammond
MICRO POWER	Rubble Trouble	24072	P.Tompkins
MICRO POWER	Swoop	31890	D.Wigg
MRM	3D Munchy	65610	M.Colson
MRM	Castle Assault	6600	K.Butler
MRM	Castle of Gems	31990	P.Orton
MRM	Crystal Castle	880	I.Melville
MRM	Q-Man	43975	B.Smith
MRM	Q-Man's Brother	8670	B.Smith

TO BE CONTINUED

Classified Ads

Rates are 18p (inc VAT) per word for members' personal ads. Members' business ads cost 30p (inc VAT) per word - and if you are selling any item in quantities rather than one-off, then that counts as business. Send cheque with order to PERSONAL ADS, PO BOX 50, ST. ALBANS, HERTS AL1 3YS. It is essential to include your membership number. Please note that we cannot arrange for box numbers.

HEAD-MASTER ROM: Prints your own headed paper with Kaga/Canon or Epson compatible printer. Any size/shape headings, with graphics and different typefaces, to your (or our) design, on ROM, called with a simple *command. Prices from £25. For details and samples send A5 SAE to HEAD-MASTER, 206 Barker Drive, St.Pancras Way, London.

Problems? try R-SOFT utilities!
1. HOW-TO: An essential collection of software and instructions for frustrated new disc owners who want to move their programs to disc.
3. ROMPULL + TAPEDUMP.
4. AUTOMATIC DISC MENU: Includes sideways-RAM version, can boot from ROM, works with 2nd 6502.
5. AUTOSOL: Will automatically boot your Solidisk with a pre-selected list of ROMs.
6. SWROM*: Puts your Basic / machine code programs in ROM format.
7. RFS-GENERATOR: Generates ROMs for the *ROM filing system. This does not use the DFS workspace and is an ideal tool to run nasty programs from disc.
All above packages £5.00 each.
8. D-MASTER: Superb disc backup program (unprotected): £7.00.
Many other utilities (BBC only - sorry!): send for our list (SAE please). R-SOFT, 22 Marriotts Close, Felmersham, Bedford, MK43 7HD. (0234) 781730.

MARKS AND STATISTICS For teachers and lecturers. Simple to use spreadsheet format. 330 students per file. Up to 300 subjects. Sorts, analyses, normalizes, calculates means, applies weighting factors. Prints histograms and lists. Full documentation supplied. Disc (40 or 80 track) £17. In-Form, 73 Woodfield Park, Colinton, Edinburgh EH13 0RA.

Acorn Teletext adaptor. Hardly used £80. Tel: (0795) 843297.

Monitor, Novex 12 inch high resolution amber screen. Composite video input, hardly used £45. Rayleigh (0268) 771368. Can transport to central London.

Cryptology. Wanted, help to convert Commodore programs emulating M 209 and Enigma to BBC. Ralph Erskine, 25 Hawthornden Road, Belfast BT4 3JU. (0232) 658066.

BBC model B, 2 * 400K Cumana DS drives, Microvitec (1431MS) colour monitor, over £150 software, all manuals, 30 discs, books, Opus organiser desk. £650 ono. S.G McDonald, 12 Portway, Wantage, Oxon. OX12 9BU.

For sale: Hewlett-Packard 41CV alphanumeric programmable scientific calculator. Complete with manuals and circuit analysis module. £150. Tel: (0562) 747923 mornings. (Worcs).

Sideways RAM for all BBC models. 32K with battery backup £40. 16K £20. No soldering - requires only one ROM socket. With write protect switch - software. SAE: Stauning (02) 288506, Udsbjergvej2, 3320 Skaevinge, Denmark.

MILLENNIUM - The wargamer's dream come true - simultaneous play on 2 BBC-B's linked via modified RS423 cable. Interstellar strategy game with 3 scenarios * 3 victory levels. Also one computer play. Disc £8.95 (state 40/80 track); with cable £12.95. Cheque/PO to: Falconsoft, P.O.Box 141, Welling, Kent DA16 2EB.

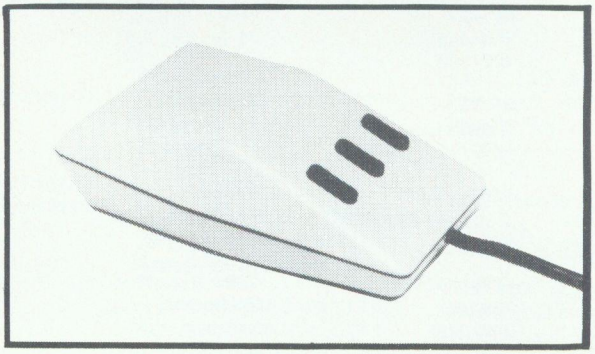
Wanted. Disc software for biorhythm charts 'B' to Star printer. Details to LKM. 74 Abingdon Road, Maidstone, Kent.

UK Bulletin Boards
(CONTINUED)

Mailbox 80 24 Hours	051 428 8924 (Liverpool) 300/300
Mailbox 83 20.00-08.30 WE 24hr	0384 635336 (West Midlands) 300/300
Maptel 24 Hours	0702 552941 (Essex) 300/300
Marctel 10.00-22.00	01 346 7150 (London) 300/300
MBBS Leconfield 24 Hours	0401 50745 (Beverley) 300/300
MBBS Mitcham 24 Hours	01 648 0018 1200/75 & 300/300
Metrotel 24 Hours	01 941 4285 (London) 1200/75 Viewdata
MG-NET Sun 17.00-22.00	01 399 2136 300/300
Microlive 24 Hours	01 579 2288 (London) 300/300
Microweb 24 Hours	061 456 4157 (Manchester) 300/300
Mileways 18.00-22.00 Mon-Fri 10.00-22.00 Sat/Sun	0533 608442 (Leicester) 300/300
MOBB 24 Hours	061 736 8449 (Manchester) 1200/75 & 300/300
Morecambe OBBS 24 Hours	0524 426133 (Morecambe) 300/300 & 1200/75
NBBS Aberdeen 24 Hours	0224 647158 (Aberdeen) 300/300
NBBS Cheshire 24 Hours	0936 77025 (Cheshire) 1200/75 & 300/300
NBBS East 24 Hours	0692 630186 (Norfolk) 1200/75 & 300/300
NBBS Essex 24 Hours	0277 228867 (Essex) 1200/75 & 300/300
NBBS London 21.00-08.00 WE 24 Hrs	01 883 5290 (London) 1200/75 & 300/300
NBBS Lutterworth 20.00-08.00 WE 24 Hrs	045 55 4798 (Lutterworth) 300/300
NBBS Midlands II 19.00-23.00 WE 24hrs	0246 865843 (Chesterfield) 300/300 & 1200/75
NBBS Marlow Sat-Thur 18.00-00.00 Ring and Ask	0628 46691 (Berkshire) 1200/75 & 300/300
NBBS NE 24 Hours	0224 641066 (Abrerdeen) 300/300 & 1200/75
NBBS Wallington 23.00-16.00	01 669 7249 (London) 1200/75 & 300/300
NKABBS 21.30-00.00	0795 842324 (Kent) 300/300 *
Norview 24 Hours	0604 20441 (Northampton) 1200/75 Viewdata
OBBS Bradford WD 18.00-00.00	0274 496783 (Bradford) 300/300
OBBS Manchester 24 Hours	061 427 1596 (Manchester) 300/300
OBBS 2 North Wales 24 Hours	0244 549336 (Wales) 300/300
Octopus 18.00-08.30	0272 421196 (Bristol) 300/300
Optel 24 Hours	01 794 0655 (London) →10 1200/75 Viewdata

MEGAMOUSE

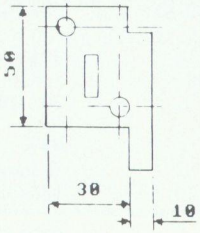
CAD
GRAPHICS



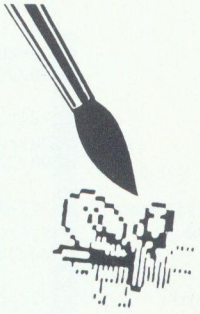
ART
DESIGN

The MEGAMOUSE hardware is a high quality UK manufactured mouse which is highly accurate and utilises a slip-free rubber-coated ball. Because of its design, quality and accuracy, it is a popular AMX replacement. £55.00 + VAT

CADMOUSE — advanced drawing package operating in Mode 0 for super resolution or Mode 1 for real time colour. Reviewed as "best drawing package on BBC", Powerful "move", "duplicate" and "invert" features are included as well as numerous CAD features. Hundreds now in use. One of few mode 0 packages on the BBC.
MEGAMOUSE and AMX Compatible £26.00 + VAT.
RB2 version £29.90 + VAT



ARTIST — dramatic new pure art package operating in Mode 2 for all 8 prime colours and hundreds of shades. Every pixel is addressable and one can design any multi-colour paint brush, air brush or icon. Innovative features include "under and over" (paint above and below other colours) "colour cycling", "colour without" (copy a picture without the background) and animation. Reviewed "unrivalled and boasts a power undreamed of". "A joy to use" etc. A+B Computers — June 1986 — "Without doubt this is the finest free hand graphics aid I have ever seen".
MEGAMOUSE and AMX Compatible £48.00 + VAT



P&P £1.50. Access, cheques, cash, POs, educational orders etc.

WIGMORE HOUSE LTD

32 Savile Row, London W1X 1AG
Tel: 01-734 0171/8826

← 6

OSI Lives! 24 Hours	01 429 3047 (London) 300/300	Stoke ITEC 24 Hours	0782 265078 (Stoke-on-Trent) 1200/75 Viewdata
Owltel 24 Hours	01 927 5820 (London) 1200/75 Viewdata	Swafax 24 Hours	0622 850440 (Kent) 1200/75 Viewdata
PBBS-NI 22.00-00.00	0762 333872 (N.Ireland) 300/300	System Aid 24 Hours	01 571 0026 (London) 1200/75 Viewdata
Pete's Place 24 Hours	0206 862354 (Essex) 300/300	TBBS Blandford 24 Hours	0258 54494 (Dorset) 300/300
PIP 24 Hours	0742 667983 (Sheffield) 300/300	TBBS London 24 Hours	01 348 9400 (London) 300/300
Pot-Bug 20.00-23.00 Mon-Fri 14.00-23.00 Sat/Sun	0782 503254 (Stoke-on-Trent) 300/300 1200/75	TBBS West Midlands 18.00-08.30	0384 635336 (London) 300/300
PSBBS 18.00-23.00	0443 755298 (Wales) 300/300	Techno Fresh systems 24 Hours	0570 423082 (Dyfed) 300/300 1200/75
React 24 Hours	0376 518818 (Essex) 300/300	Technoline 24 Hours	01 450 9764 (London) 1200/75 Viewdata
RSGB 24 Hours	0707 57477 (Hertfordshire) 1200/75 Viewdata	Telemac 15, OBBS 24 Hours	0625 33703 300/300
SABBS 24 Hours	0698 884804 (Larkhall) 300/300	Timestep (Suffolk)	0440 820002 1200/75
Sanctuary 24 Hours	0784 38110 (Surrey) 300/300	TUG London 24 Hours	01 200 7577 (London) 300/300
SBBS The Irishman 21.00-09.00	0247 455162 (N.Ireland) 300/300	TUG Board II 20.00-08.00	021 444 1484 (Birmingham) 300/300
SBBS Watford 21.00-08.00	0923 676644 (Hertfordshire) 300/300	WBBS Lowestoft 18.00-00.00	0502 515935 (Suffolk) 300/300
Southern BBS 24 Hours	0243 511077 300/300	WBBS Wimbledon 19.00-02.00	01 542 3772 (London) 1200/75 & 300/300

Discounts (CONTINUED)

Simonsoft,
25 Tatham Road, Abingdon,
Oxon. OX14 1QB

5% to members of BEEBUG on Sprites v.2, Superfruit, Castaway. 5% to members of ELBUG Sprites v.2, Superfruit.

A. Spencer,
74 Dovers Park,
Bathford,
Bath BA1 7UE

25% to members for Journal Indexing System and Objective Test System.

Thurston's Electronics Supplies Ltd.,
Thurston House,
18 Shirehall Lane,
Hendon,
London NW4 2PB

Mail Order Only. Discounts to members on some items. Phone for details.

Total Business Service,
29 Hollow Way Lane,
Amersham, Bucks.

A free program with every two purchased.

Universal Stands,
43 Pearce Avenue,
Parkstone, Poole,
Dorset BH14 8EG
Tel: 0202 740147

5% discount to members on their Universal Stand.

Weserve
128 West Street,
Porchester,
Hants. PO16 9XE

5% to 10% discount on hardware, and 13% off all Rom's/Software.

Wholesale Supplies Ltd.
Woking.
Tel: 04862 62020

Juki 6100 daisywheel printer. Model 6100. Brand new, fully guaranteed, free delivery UK mainland, £369 inc. VAT.

Zero Software,
29 St. Michaels Close,
N. Waltham,
Hampshire RG25 2BP

£1.95 off "EASYCALC" package.

ADVERTISING IN BEEBUG

For advertising details please contact:

Lynn Lloyd on (0923) 677222 or Yolanda Turuelo on (0727) 40303

or write to
P.O. BOX 50
St. Albans Herts.

LLAMASOFT

JEFF MINTER's

colourspace

A LIGHT SYNTHESISER

"So brilliant that I find words too cumbersome to describe it.. **the perfect accompaniment to any kind of music-**"

"You get to create **amazing flowing geometrical bursts of colour** on your computer-"

"..a delight to use and to watch.. a superb visual accompaniment to music.. **puts many a disco to shame.**"

"..intended to blow your mind- in the nicest possible way.. **an interactive firework display.**"

"..anyone who can manipulate a joystick or press a key can will enjoy **COLOURSPACE..**"

NOW AVAILABLE FOR the BBC B - £7.95 on tape from your computer store or direct from

Llamasoft Entertainment Software

49 MOUNT PLEASANT TADLEY HANTS (tel 07356 4478)

Movie Maker

If you fancy yourself as the next Orson Wells then this may be your chance. Geoff Bains occupies the director's chair with Movie Maker from the unlikely named Slippery Slug.

Product : Movie Maker
Supplier : Slippery Slug Software
PO Box 83, Bath, BA1 1ZA.
(0225) 69236
Price : £29.90

This unusual package, from an unusually-named company, offers BBC owners the chance to join the moguls of Hollywood and make their own mini-series.

Movie Maker is a screen design package for mode 7 that combines both static and dynamic effects. Within the limitations of the Beeb and its mode 7 display you can produce a 'film' running for several minutes (the demo supplied lasts about 20 minutes). The pack comprises two sideways ROMs, a demo cassette, and a rather poor 50 page manual.

Everything you create with Movie Maker is based on a 'scenery map' the size of 90 teletext screens. Onto this are put predefined 'shapes'. These can also form a 'cycle' - a short series of frames that are displayed one after another. The shapes are placed and moved according to a program also entered by the user.

The shapes are defined on what must be one of the best teletext screen designers around. The shapes are drawn out using a cursor the size of a single mode 7 pixel under the control of the cursor keys. Unusually these provide not only the four expected directions but diagonal movement as well when combined with the Shift key.

Colours and other teletext effects are produced with the function keys and the designer is clever enough not to let you erase the control codes when drawing. When the shape is complete it is stored away to be called up again by name at any time. In a similar manner, shapes can be converted into a cycle of frames with a simple editor and these stored as well.

When the casting is finished, the filming can begin. Using the cursor keys the shapes are placed on the scenery map in sequence as they are required. The shapes can be set moving automatically and the screen window set to follow close behind, or centre on them, or stay just where it is. Some shapes can be made to pass 'in front of' or 'behind' others.

The movie programming process is controlled with the cursor keys, a few single key commands and Movie Maker's inherent intelligence. Captions can be displayed at the bottom of the screen and even speech 'bubbles' assigned to shapes on the screen, their speech scrolling across an eleven character window above the shape. When the whole movie's in the 'can' you can save the entire film or even generate a stand alone program to show it on a Beeb not equipped with Movie Maker.

The whole process of defining shapes and assembling the movie is not a simple one, and the single key commands take a while to get used to. However, the package can produce some excellent results, as the demo 'movie' demonstrates. The program has obvious applications in the development of educational software and demonstrational/ instructional material. However, not only does Movie Maker succeed well in these fields, but it provides excellent opportunities for just having fun as well.

POLISHING PROGRAMS

Surac shows how to put some style into your programs and speed up execution at the same time.

This month we'll look at ways of polishing a program to make it both easier to understand and faster running. We will start with a program written in a very simple version of Basic, and see how we can improve things by using the features of BBC Basic.

As a starting point I will take a program that is sometimes used as a benchmark to test the speed of a micro. I will NOT be trying to criticise the original program - it was written to test lots of different computers and had to be in a fairly basic Basic. The program is a version of the "Sieve of Eratosthenes", a quick way of finding prime numbers.

Understanding the program is an exercise I will leave to you, and which demonstrates how hard it is to follow raw Basic. The program's output is the array F(), which identifies the prime numbers between 1 and 5000; locations in F() with prime indices hold "1", and non-prime indices are set to zero. Thus, at the end of a run, F(17) is set to 1 and F(18) holds 0. As a reference, this version takes 48.73 secs to run.

As a starter, let's see what we can do to speed it up. First of all it uses floating-point (FP) variables, although the program only ever deals with whole numbers.

```
        PRIME NUMBERS 1
100 S=5000
110 DIM F(S)
120 PRINT "Starting"
130 TIME=0
140 C=0
150 FOR I=1 TO S STEP 2
160   F(I)=1
170   NEXT
180 F(0)=1:F(2)=1
190 FOR I=0 TO S
200   IF F(I)=0 THEN 280
210   P=I+I+3
220   K=P+P
230   IF K>S THEN 270
240   F(K)=0
250   K=K+P
260   GOTO 230
270   C=C+1
280   NEXT
290 PRINT "Done in ";TIME/100;" secs"
```

Unfortunately, FP numbers occupy more memory - 5 bytes each in the Beeb, against 4 - and take longer to manipulate than integers.

Furthermore, F(5000) occupies 25005 bytes (Do you know why?), although each cell only ever holds 1 or 0. That's a terrible waste; BBC Basic allows "byte arrays", with only a single byte allocated to each array location. That's more than adequate for this application, as well as faster, even though only a single bit is really needed. So, using integer variables (a % at the end of the name) and a byte array gives us version 2.

Note how we now get to the array elements with the "indirection operators" "?" and "!". The running time? Just 27.29 secs.

That's much faster, but it's still hard to see what the program is doing. However, because BBC Basic is not confined to one- or two-letter variable names, we can use meaningful words, which make everything much easier to understand. For instance, "S%" holds the maximum possible size prime we are looking for - why not call it "Size%"? Don't forget, the "%" defines an integer variable.

```
          PRIME NUMBERS 2
100 S%=5000
110 DIM F% S%
120 PRINT "Starting"
130 TIME=0
140 C%=0
150 !F%=&10101
160 FOR I%=3 TO S% STEP 2
170   F%!I%=1
180   NEXT
190 FOR I%=0 TO S%
200   IF F%?I%=0 THEN 280
210   P%=I%+I%+3
220   K%=P%+P%
230   IF K%>S% THEN 270
240   F%?K%=0
250   K%=K%+P%
260   GOTO 230
270   C%=C%+1
280   NEXT
290 PRINT "Done in ";TIME/100;" secs"
```

We can also make the structure a bit clearer. Look at that listing - if you follow it through (not easy), you will see that the code at lines 210 to 270 inclusive is executed whenever F%?I% is non-zero. If that location is zero, the block of code is omitted. If we convert the code to a procedure we can pull it out of the main loop and make the action of the IF a lot clearer:

```
          PRIME NUMBERS 3
100 Size%=5000
110 DIM Flag% Size%
120 PRINT "Starting"
130 TIME=0
140 Count%=0
150 !Flag%=&10101
160 FOR Index%=3 TO Size% STEP 2
170   Flag%!Index%=1
180   NEXT
190 FOR Index%=0 TO Size%
200   IF Flag%?Index% THEN PROCCancel
210   NEXT
220 PRINT "Done in ";TIME/100;" secs"
500 END
990
1000 DEF PROCCancel
1010 Prime%=Index%+Index%+3
1020 Pter%=Prime%+Prime%
1030 IF Pter%>Size% THEN 1070
1040 Flag%?Pter%=0
1050 Pter%=Pter%+Prime%
1060 GOTO1030
1070 Count%=Count%+1
1080 ENDPROC
```

That makes what is going on rather clearer. However, mainly because of the long variable names, it takes longer to run: 40.62 secs. Note that the IF at line 200 makes use of the fact that any non-zero value is treated as TRUE.

PROCCancel, with its spaghetti GOTOs, still isn't easy to understand. Lines 1030-1060 cancel all multiples of a prime while Pter% is less than Size%. Pull that bit out as another procedure, and change its structure to make the control flow clearer:

```
1000 DEF PROCCancel
1010 Prime%=Index%+Index%+3
1020 Pter%=Prime%+Prime%
1030 IF Pter%<=Size% THEN PROCNoPrime
1040 Count%=Count%+1
1050 ENDPROC
1990
2000 DEF PROCNoPrime
2010 REPEAT
2020   Flag%?Pter%=0
2030   Pter%=Pter%+Prime%
2040   UNTIL Pter%>Size%
2050 ENDPROC
```

The test at line 1030 is reversed to allow us into PROCNoPrime. The code of the latter shows much more clearly what is happening. Easier to follow, and faster: 38.1 secs.

I used a REPEAT-UNTIL loop there, to reflect as accurately as I could the original program. However, this is not necessarily the best approach. The loop REPEATs some events and/or calculations UNTIL a pre-defined condition occurs. In this case, the repeated events are a cancellation and a simple addition of a fixed value to Pter%. The UNTIL event is that Pter% exceeds Size%. Think again; that's a FOR-TO-STEP-NEXT loop, and we could restructure PROCNoPrime as:

```
2000 DEF PROCNoPrime
2010 FOR Pter%=Pter% TO Size%
       STEP Prime%
2020   Flag%?Pter%=0
2030   NEXT
2040 ENDPROC
```

That is not only easier to follow, but it is much faster. It calculates the primes below 5000 in just 22.75 secs. The moral - choose your loops carefully.

→38

SOFTWARE FOR SIDEWAYS RAM

(Part 1)

Bernard Hill, author of BEEBUGSOFT's RomIt, looks into the creation of useful software for sideways RAM and ROM.

Although the BBC micro's facility for sideways ROMs is one of its most useful features, most users have little in-depth knowledge of the system and how to write their own sideways software. Surprisingly, ROM software is not difficult to produce, provided you have some familiarity with assembler. Even if you do not, the programs accompanying these articles are easily tailored to your own use.

Your own sideways software can be blown into EPROM with an EPROM programmer (such as that featured in BEEBUG Vol.4 Nos. 4 and 5) or you can store the image on any filing system of your choice for loading into sideways RAM. A second processor should be disabled before entering these programs although the resulting EPROM will be Tube compatible.

DIFFERENT SYSTEMS

In order to make best use of these programs you should ideally have a BBC micro with Basic II together with sideways RAM for testing any improvements or changes you will make to the programs.

Basic II users without sideways RAM will have to blow the EPROM before testing and will need to assemble it in 'normal' RAM for running at &8000. Just change the value of O% in each program (line 100) to &5C00. Basic I users with sideways RAM will need to replace the EQUS, EQUB and EQUW statements in the programs with the routines found in BEEBUG Vol.4 No.3 Page 42. Basic I users can only use these programs with sideways RAM, and cannot use them to program EPROMs.

If your sideways RAM board does not allow all writes to memory between &8000 and &BFFF to go to sideways RAM, the programs should be altered to write-enable the RAM, or to assemble at &5C00 (Basic II only) and install the code in sideways RAM later. Check the relevant manual.

BLOWING YOUR EPROM

When your ROM is complete you will wish to save it, and perhaps blow it in an EPROM. To save the software assembled in sideways RAM to the current filing system as 'ROM', use the following short program. The value of rom% and len% should be altered to suit the ROM socket number and length of ROM to be saved.

```
10 REM ROM/RAM DOWNLOADER
20 MODE 7
30 rom%=&F:len%=&2000:Y%=rom%
40 FOR I%=0 TO len%:PRINTTAB(1,1)~I%
50 !&F6=I%+&8000:I%?&3000=USR(&FFB9)
60 NEXT
70 $&900=("SAVE ROM 3000 "+STR$(len%+
1)+" 8000 8000"):X%=0:Y%=9:CALL &FFF7
```

SERVICE CALLS

Sideways ROMs are of two basic types: languages and service ROMs. Here we shall deal with service ROMs only as writing a complete language is no mean task!

At various times during operation the OS offers all the paged ROMs in your micro the opportunity to respond to particular requests. The type of request is specified by the service call number and there are many types of service call which the system issues. However, the calls do not change across the whole series of Acorn computers, from Electron to Master (although the Master series has several extra service calls).

When you press Break or turn on your micro, the OS will first evaluate where there are ROMs by looking in each socket for a recognisable header. In particular the contents of location &8006 (&82 for service ROMs) is stored in a table located (with OS 1.2) at &2A1 (ROM 0) to &2B0 (ROM 15). Thus to disable the ROM in socket n is easy, just type:

```
?(&2A1+n)=0
```

but remember that it will be enabled again with Break. To re-enable a service ROM without Break you will need to re-store &82 at the same address. This will not work with a second processor.

The service call system is very simple. The OS places the call number in the accumulator and does a JSR &8003 into each of the service ROMs in turn starting at number &F. So if your ROM is not to respond to a particular service call you must RTS with all registers intact so that some other ROM can act on it. If you do want to respond then again you must RTS afterwards with all registers restored to give other ROMs their chance. If you want to prevent them having a go then first load A with Ø before the RTS. Easy!

The complicated part is understanding why and when the various service calls are issued. To help you understand this the first program assembles a ROM header together with enough code to print out the contents of the A, X and Y registers in response to every service call which comes its way, and before the service call is passed on to the other ROMs.

When the code is assembled in sideways RAM and Break is pressed, all the service calls are reported. You will find that the accumulator (A) contains the service call number, as expected and that X is always the same (the ROM number of your sideways RAM - the currently active ROM). Y will vary. Some service calls make use of this register, some don't.

It is well worth spending some time familiarizing yourself with which service calls are issued at what times. Some of the calls you will see are:

Pressing Break	: Calls 1, 2, 3, &FE and others from DFS.
Pressing Escape	: Call 6 (error)
Disc accesses	: Call &C
*HELP	: Call 9
*commands	: Call 4 unless OS command
*TAPE OR *ROM	: Calls &F,&1Ø

The detailed meaning of these and the other calls used can be found in chapter 15 of the Advanced User Guide. In this article we shall use only calls 1 and 9 and in the future call 4.

CREATING A USEFUL SIDEWAYS ROM

In order to make your ROM well-behaved - one which does not interfere with other ROMs - it is necessary to stick to the simple rules laid down by Acorn:

A. The Rom must have a correct header. In the programs this is created by PROCromhead. It has three string parameters: a title string, version string and copyright string. Feel free to alter those listed to your own choice of name, etc. As part of this header, at &8003 the OS must find a JMP to the start of your code.
B. The code must terminate with RTS.
C. All registers must be restored to their entry value before leaving.

```
Property of:
BEEBUG Publications Ltd.
P.O.Box 50. Holywell Hill.
St.Albans. Herts. AL1 3YS.

BBC Computer
DUMPMASTER
HELP (*H)
ICON MASTER
SPELLCHECK III
TOOLKIT PLUS
Watford Electronics DFS 1.43
BASIC
>
```

Our first offering as a useful sideways ROM is shown in the second program. This should be typed in with the first program still in memory. It comprises four new procedures any of which can be omitted if you do not require a particular feature. Simply remove the procedure call from the program in lines 100-160 and the unwanted procedure definitions. Here are all the procedures you'll finish with:

PROCromhead (vital)
From Program 1. Sets up the ROM header.

PROChelp (optional)
Produces code which responds to *HELP (service call 9) by printing the ROM title. It does not do anything if the *HELP has any parameters (e.g. *HELP DFS)

PROCfxcalls (optional)
Produces code to issue *FX calls when Break is pressed (service call 1) to, say, set up your printer with *FX5,2 and *FX6,Ø or change the auto-repeat rates. The *FX call data is in DATA statements from line 10000. Put the three parameters of *FX A,X,Y on each line, padding with zeros as

required, and end with 0,0,0. The data given changes the beep to a gentle buzz, issues a *TV0 1, and enters lower case. The parameter which the procedure uses is the line number of the first DATA statement.

PROCbreakmessage (optional)
Produces code to print a message on pressing Break. Any message can be printed as defined in DATA statements from line 11000, even a full colour mode 7 logo. The DATA must terminate with an empty string and blank lines should contain a single space. The parameter passed is the line number of the first DATA line.

PROCfkeys (optional)
Sets up the function keys on Ctrl-Break. The function keys required should be set up before running this program. They are then copied into RAM and restored whenever Ctrl-Break is pressed (not on Master 128).

PROCendrom (vital)
From program 1. Provides the RTS.

The modular nature of the program makes the code in the ROM repetitive and wasteful of space. However, the source code is much easier to follow and we are very unlikely to run out of space - 8K is plenty for all we need to do!

Type in the program and be sure to save it before you run it. Mistakes very often lead to non-working ROMs and pressing Break can make the system crash, needing a switch-off! A RAM disabling switch on your ROM board becomes very useful at times like these.

Further articles in this series will explain how to write your own procedures and *commands and introduce routines to drive printers, list ROMs and re-define other ROMs' *commands to avoid command clashes.

If you would like to explore this area further we recommend Bernard Hill's RomIt from BEEBUGSOFT. This contains a full 16K of machine code and provides users with a wide range of star commands for use with sideways RAM and ROM.

```
10 REM PROGRAM SERVICE CALL DISPLAY
20 REM VERSION B0.1
30 REM AUTHOR  B.R.HILL
40 REM BEEBUG  JUNE 1986
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
100 P%=&8000:O%=&8000
110 PROCromhead("Service reporter","v1
.0","BEEBUG 1986")
120 PROCreportcode
130 PROCendrom
140 END
150 :
1000 DEFPROCromhead(name$,ver$,owner$)
1010 Q%=P%:R%=O%
1020 FOR opt=4 TO 7 STEP 3
1030 P%=Q%:O%=R%
1040 [ OPT opt
1050 EQUW 0
1060 EQUB 0        \ not a language rom
1070 JMP serve     \ service entry pt
1080 EQUB &82      \ rom type
1090 EQUB copyrite MOD 256
1100 EQUB 0
1110 .title    EQUS CHR$13+name$
1120 .verstr   EQUS CHR$0+ver$
1130 .copyrite EQUS CHR$0+"(C) "+owner$
+CHR$0
1140 .serve
1150 ]:NEXT opt
1160 ENDPROC
1170 :
2000 DEFPROCreportcode
2010 Q%=P%:R%=O%
2020 FOR opt=4 TO 7 STEP 3
2030 P%=Q%:O%=R%
2040 [ OPT opt
2050 PHA
2060 TXA:PHA
2070 TYA:PHA      \ save registers
2080 LDX #0
2090 .loop
2100 LDA mess,X   \ print message
2110 BEQ endmess \ 0 marks message end
2120 JSR &FFE3    \ osasci- write byte
2130 INX
2140 JMP loop
2150 .endmess
2160 TSX:LDA &103,X \ get A off stack
2170 JSR printbyte
2180 TSX:LDA &102,X \ get X
2190 JSR printbyte
2200 TSX:LDA &101,X \ get Y
2210 JSR printbyte
2220 JSR &FFE7    \ osnewl - print CR
2230 PLA:TAY
2240 PLA:TAX
2250 PLA          \ restore registers
2260 JMP finish   \ jump to end
2270 .mess        \ data
2280 EQUS "Service call: A,X,Y="
2290 EQUB 0
2300 .printbyte
2310 PHA          \ save for later
```

```
2320 LDA #32
2330 JSR &FFE3  \ write a space
2340 PLA        \ restore A
2350 PHA        \ but resave for later
2360 LSR A      \ find
2370 LSR A      \      top
2380 LSR A      \         half
2390 LSR A      \            byte
2400 JSR outbyte \ output to screen
2410 PLA        \ old value
2420 AND #&F    \ find low half-byte
2430 .outbyte   \ and output to screen
2440 TAX
2450 LDA nums,X  \ to ascii
2460 JSR &FFE3   \ print
2470 RTS
2480 .mess      \ data area
2490 EQUS "Service call : A,X,Y="
2500 EQUB 0     \ zero terminator
2510 .nums
2520 EQUS "0123456789ABCDEF"
2530 .finish
2540 ]:NEXT opt
2550 ENDPROC
2560 :
9000 DEFPROCendrom
9010 [ OPT 7:RTS:]
9020 ENDPROC
```

```
  10 REM PROGRAM SIDEWAYS ROM
 100 P%=&8000:O%=&8000:S%=O%
 110 PROCromhead("Custom Rom","V1.0","B
EEBUG, 1986")
 120 PROChelp
 130 PROCfxcalls(10000)
 140 PROCbreakmessage(11000)
 150 PROCfkeys
 160 PROCendrom
 170 END
 180 :
2000 DEFPROChelp
2010 Q%=P%:R%=O%
2020 FOR opt=4 TO 7 STEP 3
2030 P%=Q%:O%=R%:REM for pass 2
2040 [ OPT opt
2050 CMP #9         \ *HELP call?
2060 BNE nohelp     \ no, exit quick
2070 PHA
2080 TXA:PHA      \ save registers
2090 TYA:PHA      \ Y too
2100 .again
2110 LDA (&F2),Y    \ end of command
2120 CMP #32        \ is it space?
2130 BNE notspace   \ .. no
2140 INY            \ yes .. find end
2150 JMP again
2160 .notspace
2170 CMP #&0D       \ is it CR?
2180 BNE endhelp    \ no so not respond
2190 LDX #0         \ initialize loop
2200 .helploop
2210 LDA &8009,X    \ get title
2220 BEQ endtitle   \ zero terminator
2230 JSR &FFE3      \ osasci - print
2240 INX            \ next title char
2250 JMP helploop
2260 .endtitle
2270 JSR &FFE7      \ osnewl - print CR
2280 .endhelp
2290 PLA:TAY      \ restore
2300 PLA:TAX      \         registers
2310 PLA
2320 .nohelp
2330 ]:NEXT
2340 ENDPROC
2350 :
3000 DEFPROCfxcalls(dataloc)
3010 Q%=P%:R%=O%
3020 FOR opt=4 TO 7 STEP 3
3030 P%=Q%:O%=R%
3040 [ OPT opt
3050 CMP #1    \ break pressed?
3060 BEQ break    \ yes
3070 JMP nobreak \ no, quick exit
3080 .break
3090 PHA        \ save
3100 TXA:PHA \     registers
3110 TYA:PHA
3120 LDA &A8:PHA  \ need two zpg
3130 LDA &A9:PHA  \   locations too
3140 LDA #fx MOD 256
3150 STA &A8    \ load pointer to
3160 LDA #fx DIV 256
3170 STA &A9    \         fx calls
3180 .dofx
3190 LDY #0
3200 LDA (&A8),Y \ get fx number
3210 BEQ finifx  \ ends with 0
3220 PHA         \ just save A a while
3230 INY
3240 LDA (&A8),Y \ 1st fx parameter
3250 TAX         \     into X
3260 INY
3270 LDA (&A8),Y \ 2nd fx parameter
3280 TAY         \     into Y
3290 PLA         \ 1st param restored
3300 JSR &FFF4   \ do *FX
3310 LDA &A8     \ now update
3320 CLC         \    pointer
3330 ADC #3      \       by
3340 STA &A8     \         adding 3
3350 BCC dofx    \ no carry, reloop
3360 INC &A9     \ carry. INC hi-byte
3370 JMP dofx    \ now do next *fx
3380 .finifx     \ all *fx are done
3390 PLA:STA &A9 \ restore
3400 PLA:STA &A8 \   everything
3410 PLA:TAY   \       which
3420 PLA:TAX   \         was
```

```
3430 PLA          \           used
3440 .nobreak
3450 JMP exitfx
3460 .fx
3470 ]:P%=P%+30:REM fx data here
3480 O%=O%+30
3490 [ .exitfx:]:NEXT opt
3500 REM now put data in place
3510 c=O%-P%
3520 RESTORE dataloc
3530 REPEAT
3540 READ a,x,y
3550 fx?c=a:c=c+1
3560 fx?c=x:c=c+1
3570 fx?c=y:c=c+1
3580 UNTIL a=0
3590 ENDPROC
3600 :
4000 DEFPROCbreakmessage(dataloc)
4010 Q%=P%:R%=O%
4020 FOR opt=4 TO 7 STEP 3
4030 P%=Q%:O%=R%
4040 [ OPT opt
4050 CMP #1         \ break pressed?
4060 BNE notbreak   \ no
4070 PHA            \ yes, so
4080 TXA:PHA      \ save all
4090 TYA:PHA      \   registers, and
4100 LDA &A8:PHA \     2 zpg bytes
4110 LDA &A9:PHA \       too
4120 LDA #message MOD 256
4130 STA &A8        \ save pointer to
4140 LDA #message DIV 256
4150 STA &A9        \        message
4160 LDY #0         \ start loop
4170 .messloop
4180 LDA (&A8),Y    \ get message byte
4190 BEQ endmess    \ ends in &00
4200 JSR &FFE3      \ osasci - write
4210 INY            \ bump pointer
4220 BNE messloop   \ not over 255 bytes
4230 INC &A9        \ over page boundary
4240 JMP messloop   \ and loop back
4250 .endmess
4260 JSR &FFE7      \ osnewl - send CR
4270 PLA:STA &A9 \ restore
4280 PLA:STA &A8 \   all values
4290 PLA:TAY      \     and
4300 PLA:TAX      \       registers
4310 PLA
4320 .notbreak
4330 JMP exitmess
4340 .message
4350 ]:P%=P%+1024:REM 1K - enough for
4360 O%=O%+1024:REM full MODE 7 screen
4370 [ .exitmess:]:NEXT opt
4380 REM now put message in rom
4390 m=message+O%-P%
4400 RESTORE dataloc
4410 REPEAT
4420 READ mess$
4430 $m=mess$
4440 m=m+LENmess$+1
4450 UNTIL mess$=""
4460 ?(m-1)=0
4470 ENDPROC
4480 :
5000 DEFPROCfkeys
5010 Q%=P%:R%=O%
5020 FOR opt=4 TO 7 STEP 3
5030 P%=Q%:O%=R%
5040 [ OPT opt
5050 CMP #1       \ break?
5060 BNE exitfkey
5070 PHA          \ save registers
5080 TXA:PHA
5090 TYA:PHA
5100 LDA #&FD     \ first find out
5110 LDX #0       \   whether we had
5120 LDY #&FF     \      hard or soft
5130 JSR &FFF4    \        break
5140 CPX #0       \ soft?
5150 BEQ finload \  yes - don't load
5160 LDX #0
5170 .fkeyloop
5180 LDA &9F00,X \ get byte from saved
5190 STA &B00,X  \ and store in fk buff
5200 INX
5210 BNE fkeyloop \ loop if not yet 256
5220 .finload
5230 PLA:TAY    \ restore all
5240 PLA:TAX    \    registers
5250 PLA
5260 .exitfkey
5270 ]:NEXT opt
5280 REM copy from current FK buffer
5290 FOR I%=0 TO &FC STEP 4
5300 I%!(&1F00+S%)=I%!&B00
5310 NEXT I%
5320 ENDPROC
5330 :
10000 REM *FX calls on break:
10010 DATA 211,4,0:REM alter beep
10020 DATA 212,184,0
10030 DATA 213,2,0
10040 DATA 214,2,0
10050 DATA 202,48,0:REM lower case
10060 DATA 144,0,1:REM *TV0 1
10070 DATA 0,0,0:REM end record
10080 REM Breakmessage
11000 DATA "Property of:"
11010 DATA " "
11020 DATA Mr Beebug Member
11030 DATA Anytown
11040 DATA Anyplace
11050 DATA ""
```

HARDWARE ANIMATION

Although animation techniques have been extensively covered in past issues of BEEBUG, Alan Webster and Simon Tresman have produced a novel and unusual approach using the Beeb's hardware.

This short article describes an unusual technique for animation based on the Beeb's hardware scrolling facility. A variety of interesting and varied effects can be produced and the results can be quite stunning. The basis of the program is the hardware scroll of the BBC's 6845 CRTC (Cathode Ray Tube Controller). This is the chip that controls most aspects of the Beeb's screen display.

Registers 12 and 13 of this chip point to the screen memory location that forms the start of the screen display. By changing the values contained in these two registers, we can cause the whole screen to shift at great speed.

To use this to animate any object on the screen, it is necessary to display the object in successive orientations and positions up the screen. Scrolling the screen display makes the shape appear to remain in the same screen positions but sequence rapidly through its different orientations. In this way shapes can be made to rotate, or shrink and grow, very rapidly. Many other interesting visual effects can be found through a process of trial and error.

The main routine is PROCscroll(N%), where the value of N% determines by how much registers 12 and 13 are to be incremented each time. This process is handled by the two VDU instructions in lines 1040 and 1050.

In a 20K mode (0, 1 or 2), each character line is made up of &280 bytes. Thus to scroll the screen by one line, the two registers should be incremented by &280. If we wish to scroll the screen sideways, a number that is not a multiple of &280 should be used.

Be careful, when scrolling the screen, that the address written to the 6845 is a valid screen address (&3000 to &8000 for a 20K mode) or unexpected effects may occur.

The short demo included with PROCscroll displays a pattern of polygons which appear to rotate through the use of hardware scrolling. Just specify the size (try 3 or 4) and the number of sides (between 2 and 6 is best) when prompted. One side will produce a dot, two sides a straight line, three sides a triangle etc. Experimenting with this program will reveal the potential of this technique, while further animation can be introduced using the more usual VDU19 method. For more details of the 6845 registers see the Advanced User Guide page 359.

```
  10 REM PROGRAM Freaky
  20 REM VERSION B0.1
  30 REM AUTHORS Alan Webster
  40 REM         Simon Tresman
  50 REM BEEBUG  JUNE 1986
  60 REM PROGRAM SUBJECT TO COPYRIGHT
  70:
 100 MODE 1:ON ERROR GOTO 100
 110 INPUTTAB(10,10)"Sides "S%
 120 INPUTTAB(4,14)"Size (1=small,6=lar
ge) "T%:IFT%<1 OR T%>6 THEN 100 ELSE T%=
(2^T%*4)
 130 CLS:PROCdraw(T%):PROCscroll(T%*40)
 140 END
 150:
1000 DEFPROCscroll(N%):B%=&800
1010 REPEAT
1020 FOR A%=&3000 TO (&8000-N%) STEPN%
1030 *FX19
1040 VDU23,0,12,A% DIV B%,0;0;0;
1050 VDU23,0,13,A% MOD B% DIV 8,0;0;0;
1060 NEXT:UNTIL INKEY-1
1070 ENDPROC
1080:
1090 DEFPROCdraw(C%)
1100 FOR X%=C% TO 1280 STEP C%*2
1110 GCOL 0,RND(3)
1120 FOR Y%=C% TO 1024-C% STEPC%*2
1130 PROCshape(X%,Y%,Y%/T%/5)
1140 NEXT,
1150 ENDPROC
1160:
1170 DEFPROCshape(M%,N%,A)
1180 S=SIN(A)*T%+M%:T=COS(A)*T%+N%
1190 MOVE S,T
1200 FOR G=A TO A+6.3 STEP (6.28/S%)
1210 S=SIN(G)*T%+M%:T=COS(G)*T%+N%
1220 DRAW S,T:NEXT
1230 ENDPROC
```

1st course

Using the ADVAL Function

If you thought ADVAL was all to do with the analogue port and hardware then think again. Mike Williams shows that there's much more to this frequently overlooked function.

Like many other features of BBC Basic, the ADVAL function has many more uses than you might expect. Certainly the main use of ADVAL is to allow Basic programs easy use of the Analogue to Digital (A to D) converter, but even if you have no intention of using this interface the ADVAL function still has much to offer as we shall see.

First of all, though, let's just run through the more obvious uses of the ADVAL function before looking at some of the less obvious applications. Many people buy joysticks (which connect to the analogue port) to use with commercial games, and it is quite easy to write routines that will allow your own programs to be controlled by joysticks where appropriate. We'll have a look at a couple of typical examples and you should be able to adapt these to suit any other requirements.

Most joysticks have some form of lever that you can move to indicate up/down and left/right, and a 'fire' button. The value you use with the ADVAL function will determine its function as shown below:

```
ADVAL(0)    read 'fire' status
ADVAL(1)    read joystick1 left/right
ADVAL(2)    read joystick1 up/down
ADVAL(3)    read joystick2 left/right
ADVAL(4)    read joystick2 up/down
```

You may sometimes find a joystick which is connected differently. You can sort things out with the first (short) program.

If you have only one joystick then set the limit of the loop to 2 in line 130. The fire button is easy and this is the first value displayed. The possible values are as described in the User Guide:

```
0 no button pressed
1 left joystick (1) button pressed
2 right joystick (2) button pressed
3 both buttons pressed
```

Experiment by moving each joystick in turn up and down, and left and right. You should see the corresponding values changing on the screen. This is where you need to know about the two kinds of joystick on the market. These are the 'switched' joystick and the 'analogue' joystick, though both types are normally connected to the Beeb's analogue port.

```
100 MODE 7:VDU23,1,0;0;0;0;
110 REPEAT
120 PRINTTAB(12,10)"ADVAL(0): ";ADVAL(
0) AND3
130 FOR I=1 TO 4
140 A$=STR$(ADVAL(I)/16)
150 PRINTTAB(12,12+I)"ADVAL("+STR$(I)+
"): ";SPC(4-LEN(A$));A$
160 NEXT I
170 UNTIL FALSE
180 END
```

The analogue type of joystick provides continuous movement from one end of its range to the other (up and down or left and right). Numerically, the values displayed by the above program will vary continuously between 0 and 4095. A program can easily use this data to control the position of any object on the screen, such as a cursor, target cross-wires etc. Note that the value obtained by the ADVAL function is divided by 16. The reason for this is explained quite clearly in the User Guide and this conversion should always be used.

The other type of joystick can only indicate direction, either up/down or left/right. Using the above program you will find a switched joystick generates only the values 0, 2048 and 4095 (or close approximations to these). These values can then be used to indicate up/down or left/right, but it is up to the software to determine a position on the screen.

SWITCHED JOYSTICK

Let's look at a short program for use with a single switched joystick. This is listed on the opposite page.

SWITCHED JOYSTICK ROUTINE

```
 100 MODE 4:VDU23,1,0;0;0;0;
 110 ON ERROR MODE 3:REPORT:END
 120 X1=640:Y1=512:X2=640:Y2=512
 130 GCOL3,1:VDU5:exit%=FALSE:INC=32
 140 MOVEX1,Y1:PRINT"A"
 150 REPEAT
 160 fire=FNmove
 170 IF fire AND 1 THEN PROCfire(X1,Y1)
 180 T%=INKEY(2)
 190 UNTIL exit%:VDU4
 200 END
 210 :
1000 DEF FNmove
1010 MOVEX1,Y1:PRINT"A"
1020 X=ADVAL(1)/16-2048:Y=ADVAL(2)/16-2
048
1030 IF X>1024 X1=X1-INC ELSE IF X<-102
4 X1=X1+INC
1040 IF Y>1024 Y1=Y1+INC ELSE IF Y<-102
4 Y1=Y1-INC
1050 MOVEX1,Y1:PRINT"A"
1060 =ADVAL(0) AND 3
1070 :
1100 DEF PROCfire(x,y)
1110 exit%=TRUE
1120 ENDPROC
```

We must accept that with devices like this there may be a little variation in the resulting values. To accommodate this, the program converts the value from the joystick (initially in the range 0 to 4095) to a number between -2048 and +2048. A value close to +2048 will indicate a move up (or right), a value close to -2048 will indicate a move down (or left), and a value close to 0 will indicate no change of direction.

In the program, the distance to move (in graphics co-ordinates) is assigned to the variable INC (32 in the listing at line 130). Larger values will produce faster movement, smaller values slower movement. The general principle of the routine is to execute repeatedly a loop, reading the information from the joystick and adjusting the position (determined by X1 and Y1) of some object displayed on the screen (in this case just a letter A). Using GCOL 3, the object is repeatedly redrawn on the screen and moves in the direction indicated by the movement of the joystick. In practice, you are also likely to need to make sure that your moving image remains within the screen area.

This can be done by testing to make sure that the values of X1 and Y1 remain within the screen limits before incrementing their values for the new position. If you want to include such a test, modify the program above by typing in the following lines:

```
1030 IF X>1024 AND X1>=INC X1=X1-INC ELSE
     IF X<-1024 AND X1<1280-INC X1=X1+INC
1040 IF Y>1024 AND Y1<1024-INC Y1=Y1+INC
   ELSE IF Y<-1024 AND Y1>INC Y1=Y1-INC
```

If you find that the 'A' moves in the opposite direction to that indicated by the joystick, then change +INC to -INC or vice versa in lines 1030 and 1040 until screen and joystick match. If you have added the revised versions of lines 1030 and 1040 then you will have to change these even further (swapping the two tests on X1 and the two tests on Y1).

The routine also reads the state of the fire button. If pressed, this calls a routine PROCfire, which in this case just causes the program to terminate. It would be quite easy to take such routines as FNmove and PROCfire and expand these to produce a joystick-driven program.

ANALOGUE JOYSTICK ROUTINE

```
 100 MODE 4:VDU23,1,0;0;0;0;
 110 ON ERROR MODE 3:REPORT:END
 120 X1=640:Y1=512:X2=640:Y2=512
 130 GCOL3,1:VDU5:exit%=FALSE
 140 PROCplot
 150 REPEAT
 160 fire=FNmove
 170 IF fire AND 1 THEN PROCfire(X1,Y1)
 180 IF fire AND 2 THEN PROCfire(X2,Y2)
 190 UNTIL exit%:VDU4
 200 END
 210 :
1000 DEF FNmove
1010 PROCplot
1020 X1=1280-ADVAL(1)/51.2:Y1=ADVAL(2)/
64
1030 X2=1280-ADVAL(3)/51.2:Y2=ADVAL(4)/
64
1040 PROCplot
1050 =ADVAL(0)AND3
1060 :
1100 DEF PROCfire(x,y)
1110 exit%=TRUE
1120 ENDPROC
1130 :
1200 DEF PROCplot
1210 MOVEX1,Y1:PRINT"A"
1220 MOVEX2,Y2:PRINT"B"
1230 ENDPROC
```

ANALOGUE JOYSTICKS

The second example is a similar routine and demonstration for a pair of analogue joysticks (the standard Acorn type). The routines used are very similar to those used before except that the value obtained from the ADVAL function is used directly to give a screen position. A further procedure PROCplot just simplifies the display of the two selected objects, in this case the letters 'A' and 'B'. Because two joysticks are in use, both fire buttons are tested in turn.

In the program above, the x co-ordinates have been adjusted (by sub-tracting the ADVAL value from the screen width 1280) to ensure that screen and joystick movement match up horizontally. The values returned by the ADVAL function have this time been divided by 51.2 in the horizontal direction and 64 in the vertical direction. This gives a range of values from 0 to 1024 vertically, and 0 to 1280 horizontally (matching normal screen graphics co-ordinates) rather than the previous 0 to 4096.

With analogue joysticks, the speed of movement about the screen is determined by your dexterity with the joystick, you can very quickly move from one side of the screen to the other. If necessary, you can use an analogue joystick like a switched joystick by using the other routines. Then the movement of the joystick merely indicates direction. However, you CANNOT make a switched joystick behave like an analogue one.

PRINTER CHECK

Finally, this month, let's have a quick look at a quite different use of the ADVAL function, one which enables a program to determine for itself whether you have switched your printer on or not. Here's the routine incorporated in a short demonstration:

```
100 MODE 7:VDU23,1,0;0;0;0;
110 IF FNtestprinter THEN PRINT"Printer operational" ELSE PRINT"Printer switched off"
120 END
130 :
1000 DEF FNtestprinter
1010 VDU2,1,0,1,0,3
1020 =(ADVAL(-4)=63)
```

Run this program with your printer first switched on, and then again with it switched off, and note the difference. If you don't have a printer then the routine will always give the same result. The function FNtestprinter can be easily incorporated in any of your own programs and enables any program to check that a printer is switched on and operational before trying to send any real output.

Next month we'll take a look at this in much more detail to see how it works. We will also look at other interesting uses of the ADVAL function.

—29 So there you have many of the keys to using BBC Basic to write clear, fast and well-structured programs. To summarise:

+ Don't use FP variables if integers will do (but remember that all the % signs use up memory space).

+ Match an array's type to what it has to hold - it costs time and memory to be able to store 5.1287E23 when you only need 1 or 0.

+ Use meaningful variable names.

+ Show program structure by making conditional blocks of code into procedures. Procedures are not limited to bits of code needed at several places in a program.

+ Impose structure on spaghetti by suitable loops.

These principles apply, with one exception, if you are trying to get the maximum speed out of a program. The Beeb's resident integer variables (A%-Z%) are MUCH faster than any others, no matter how meaningful. As an example, I took the final version of the program above and went back to the resident variables of the second listing - it ran in a mere 14.59 secs. However, unless speed is vital, it's normally better to use longer names; you'll notice the difference when you go back to a program after 6 months!

Remember, too, that there is no substitute for a good and efficient algorithm in the first place. Time spent here will often produce better results than trying to speed up a poor algorithm. For example, there is no point checking all even numbers when searching for primes as 2 is the only even prime number.

Macro Assemblers

The latest software tools for machine code programmers are macro assemblers. Geoff Bains has been taking a look.

Giving the Beeb a built-in assembler was without doubt a masterful move on the part of Acorn. However, this primitive assembler offers only the bare essentials. More comprehensive assembler packages suitable for serious use have now been produced by several other companies. Here we look at two of these.

The major difference with these new assemblers is that they can handle 'macros' - sections of code that can be defined as a macro and called up and inserted into the machine code generated (the 'object code') whenever needed. It is a similar idea to subroutines except that the macro code itself is actually inserted in place of the call.

In addition, these newer assemblers also have the facility to assemble from source code that is far larger than the Beeb's memory. This is especially useful if you are, say, assembling a ROM. What with the bulky mnemonics and extensive comments, the source code for an 8K ROM may take up around 50K. Clearly, this calls for a new assembler.

Product : Macrom
Supplier : Clares Micro Supplies
98, Middlewich Road,
Northwich, Cheshire, CW9 7DA.
(0606) 48511
Price : £40 (ROM) £35 (disc)

Macrom is a very powerful assembler package combining a text editor, macro assembler, disassembler, and assembler routine library.

The Macrom editor is a fully fledged text editor in its own right, though it is especially configured for writing assembler code. The editor operates in any display mode and is entered with either the command *ED or *ED <filename> to edit a file already saved to disc or tape.

The assembler text is entered as normal into the editor. A string starting with a space is assumed to be an op code. Strings without are taken as labels. As soon as Return is pressed at the end of a line the editor automatically formats the line into its four fields of label, op code, operand and comment.

Needless to say you can use the cursor keys to go back on existing lines and alter them with the utmost ease. The function keys give you a range of facilities that owe more than a little to Wordwise. The editor can be toggled between overwrite and insert modes and a 'move to' command is also provided. Markers can be inserted in the listing and the blocks deleted, moved or copied.

The equivalent of Wordwise's menu page is provided by 'extended' commands entered in the window that springs up at the screen base when f1 is pressed. The extended commands will display the assembler listing (as Wordwise's preview), load and save blocks or the whole listing, change display mode, perform a search and replace operation, and assemble the code.

```
; This macro sets up the header for
; a sideways ROM
; requires  1/ language entry
; 2/ service entry
; 3/ ROM type
; 4/ binary version no.
; 5/ ROM title string
; 6/ version string
; 7/ copyright message

SWR         MACRO
offset      DEFL Copy-origin
origin      JMP &1
            JMP &2
            DEFB &3
            DEFB offset
            DEFB &4
            DEFM &5
            DEFB 0
            DEFM &6
Copy        DEFB 0
            DEFM "(C)"
            DEFM &7
            DEFB 0
            ■
```

The clever part of the Macrom editor is that the assembler is not stored as ASCII text. It is tokenized so that you can cram much more assembler source code into the Beeb's modest memory. Pure ASCII files (from, say, Wordwise) can be loaded into the editor using *EXEC. In addition, a program supplied on the support disc

will convert Basic programs containing assembler into Macrom editor files.

Once your assembler program, or macro is written, it should be assembled. Macrom's assembler is extremely fast and provides for a number of options that control the assembly. The object code can be directed to a file on disc or tape, or directly into memory. The assembly listing can be printed or not as required and indeed the actual assembly output can be suppressed for a fast run through to detect errors. The assembler can suppress errors, stop at an error and print the offending line, or direct all the errors to a file for inspection after assembly.

There's not a lot more you could ask of an assembler. Macrom not only performs all these tasks but does so efficiently and quickly. The real proof of the pudding, however, has to be the command set allowed by the assembler.

Most of the normal assembler mnemonics and operators are allowed and several special ones as well. Op codes are only accepted in upper case leaving lower case free for use as labels. The extra op codes for the 65C02 (as used in the B+, Master, and 65C02 second processor) are also accepted but a warning is printed on assembly in case they are not wanted.

For Beeb directives such as EQUB, the assembler not only allows the Basic form but other commonly used mnemonics like DEFB, DB, and DFB as well. Unfortunately Macrom does not allow the operator DIV so if you're used to using this to extract the high byte from a number, you'll have to get accustomed to the expressions '<' and '>' which are used to specify the high and low bytes of a number.

Macrom can support conditional assembly too. The directives IF, ELSE, and ENDIF are used to alter the flow of assembly at assembly time according to factors under your control.

Other pseudo-ops control the assembly. ORG and RUN define the start and execution addresses of the object code and EMBED sets the relocation address for code to be downloaded at run-time. A very useful directive, QUERY, will halt the assembly and allow a value to be entered by the user. The most important pseudo-ops supported by the package, however, are those that instigate macros.

A section of code that is to be a macro is defined within the pseudo-ops MACRO and ENDM. The label on the assembler line containing MACRO is the title of the macro and is used to call the code from the main assembler listing. Alternatively CHAIN "<filename>" will switch the assembler source from the main listing to the specified file on disc, to access other listings and other macros.

Parameters can be passed to the macro to affect the exact nature of the code inserted at that point. Labels and values following the call to the macro are substituted in order for @1, @2, etc in the macro code. With the facility for conditional assembly, this provides a very powerful combination. The most complex assembler program can be simply broken down into easily-written chunks.

To take advantage of the macro facility of Macrom, Clares also supplies a library of useful macros on disc. These cover a wide range of subjects from changing mode and printing characters to setting up a ROM header.

The Macrom ROM also includes a dis-assembler, invoked through *PEEK. It will only operate on code in memory but it does have the useful facility of displaying the full names of all OS calls and vectors.

Macrom is a very powerful package that will appeal to any serious assembler programmer. It has been designed carefully to fit in with existing ideas of the BBC Basic assembler but offers facilities far beyond that. The manual is well written and acts as both a clear, concise reference and a helpful tutorial. Full marks for an excellent product.

Product : 6502 Macro Assembler
Supplier : Star Software
PO Box 5, Wigan,
Lancashire, WN5 7QB.
Price : £24

Star's macro assembler is a different kettle of fish to Clares'. It does not offer the same range of features as Macrom but then it costs a lot less too.

Star's assembler comes on a 8K ROM which comprises only an assembler. This ROM has no editor or disassembler, but then these may be found in many other packages.

The Star assembler does not require an editor for writing the source code because every user of this ROM already has one. The Star assembler uses files created with the BBC Basic interpreter. That is not to say that the square brackets, OPT, loop structures, and other paraphernalia required by the Beeb's built-in assembler are necessary. Instead, an assembler listing is created as a series of program lines containing assembler statements.

This means that the source code can be created just as you create a normal Basic program. It can be edited, renumbered, and so on as normal and saved and loaded from disc or cassette in the usual way too. If you have a Basic program editor of some kind then this can be used to supplement BBC Basic's editor to good effect.

```
*AUTO

*LIST
   10 \ printing strings
   20 .M1 "Beebug":#def L1 .-M1
   30 .M2 "for the BBC micro":#def L2 .-
M2
   40
   50 #define out_chr &ffee
   60
   70 \macro to print string at "addr"
   80 #macro show_string(addr,len)
   90 LDX #0
  100 .loop LDA addr,X:JSR out_chr
  110 INX:CPX #len:BCC loop
  120 JSR &FFE7
  130 END
  140
  150 show_string(M1,L1)
  160 show_string(M2,L2)
  170 RTS
*
   50 #define out_chr &FFEE
*ASSEMBLE
```

However you edit your source code, it is assembled with Star's assembler with the command *ASSEMBLE. This prompts you for the filename of the file containing the program or '*' can be entered instead to assemble the program in memory.

Star's assembler does not assemble at quite the speed of Macrom but that's not to say that it's in any way slow. The object code generated is placed in an 8K output buffer in memory on its way to a file on disc or cassette. The object code is always saved with the same name - 'FILEa'. If this gets to be too big (more than 8K) a second file - 'FILEb' - is created and so on. Up to around 50K of object code can be generated in this way.

Like Macrom, Star's assembler supports all the normal 6502 op codes and has several others with special functions. It can also support the 65C02 extra op codes but only after a #new command in the source code. This is in many ways a safer solution to the problem than that employed by Macrom.

All of the Star assembler's pseudo-ops - to set the start and execution address, switch the printer on and off, and so on - are accessed with these # commands. This makes the assembler much easier for the beginner to come to grips with - the program code itself and the controlling commands are clearly separated.

This assembler can also support conditional assembly with #if, #else, and #endif commands. Rather disappointingly, the functions EQUB, EQUS, and so on are not accessed with these names in Star's assembler. Instead, EQUB is called #byte and EQUW is #word. EQUS is called up rather confusingly with just ". However, a nice extension to this theme is the #include command that will place an entire file of codes into memory at that point in the object code - great for inserting large amounts of data prepared beforehand.

To handle very large source code programs, split over several files, the #merge command will make the assembler treat the files as one. Macros are defined and used in Star's assembler in a very similar way to Macrom. The #macro command starts the definition and END finishes it. The macros can then be called up ('expanded') by name whenever required. Parameters can be passed to the macros too, in a very similar manner to parameter passing to procedures in Basic.

Although the Star package is not as comprehensive as Macrom it still offers a remarkably powerful program for the price. The manual is clear and helpful and has many example listings to show the use of this ROM's numerous facilities. Star's assembler is excellent for anyone just starting on the road to serious assembler work or who does not require the extra facilities of Macrom.

BEEBUG 29 FEB 1986 A.I. POSTBAG

Filer in a Tight Spot

I very much liked the concepts, style and structure of the database program, Filer, published in three parts in BEEBUG Vol.4 Nos.6 to 8. It seemed to be just what I'd been looking for. I wanted to use it with my ADFS, so the comment about the program being able to run with PAGE at its default value was reassuring that this would be possible, and I duly typed it in.

After minimal debugging I was able to create, display, sort and extract records as required. Imagine my disappointment when after a few test selections, sorts and what-have-you I got the dreaded "No room" error. I shortened the program as much as possible but I still get "No room" errors after a short period of use.

As if that wasn't enough, a SORT on the 14 records in my test datafile took 2 minutes and 9 seconds.

Whatever the shortcomings of the program, it certainly stimulated my interest, as do many of the articles you publish.

Stuart W. Moore

We have had to summarise Mr Moore's detailed letter which contained much useful and interesting comment. With the ADFS on the B+ (and on the Electron) it is not possible to reset PAGE below &1900. Although Filer will run with this value of PAGE, it does depend on your own data, and the original article did say that the program was best run with PAGE at &1400.

The program can be shortened by removing unnecessary spaces, reducing long variable names and similar measures. Further compaction can be achieved by using a compacting routine like the *PACK in Toolkit. More drastic reductions can be made by dividing the program up into separate modules, e.g. one for file management, one for record management and one for sorting. This is quite easy to do and we have one version of Filer working in this way.

The sort used is relatively slow (stated in the magazine), but reasonable for many uses of Filer. By separating the sort from the rest of the program a much more efficient technique could be implemented. Thanks, too, to all the many other members who have written in with useful comments and suggestions on Filer.

Master Software

I would like to clear up a small area of confusion which seems to have arisen over the terminal software in the Master 1Mbit ROM. In a number of places I have seen it stated that the 'Termulator' ROM is included. This is not so. What you get is a small (2K) 'terminal' program which offers limited VT100 emulation. Termulator is an entirely different product available separately on a 16K ROM.

On the same note, the editor which is included is not 'Basic Editor' but is an enhanced version of 'EDIT' from the 6502 Development Pack.

Paul Fellows
Acorn

We are happy to put the record straight. Many of the early reports on the Master series contained some inaccuracies of this kind.

Blackjack Scoops the Pool

I'd like to thank you and Messrs P.Jacobs and J.Button for the Blackjack game on the cassette for Vol.4 No.9. As far as I can see it is perfect in design and execution, and has excellent strategy. My only complaint is that it is wasting hours of my time in playing it. Please convey my appreciation to the authors.

Sebastian Lazereno

Alan Thorpe was another who wrote praising this game. He asks if the method of scrolling backwards and forwards through the instructions could be described in an article in BEEBUG. We have asked the authors to consider this. Copies of the relevant cassette/disc are still available for those who would like to obtain this game - see inside back cover of the magazine for details.

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINT and

Basic I Relocation

Basic I users can imitate the relocation facility of the assembler in Basic II by using an offset whenever an absolute address is used. For example:

```
100 FOR pass=0 TO 3 STE
P 3:P%=&2000:O%=&6000
110 [OPT pass
120 JMP address+O%
130 JSR location+O%
140 LDA index+O%,X
etc.
```

To relocate the assembled code now only requires the alteration of the offset, O%.

Cedric Marshall

Masking

Whereas DIV and MOD can be used to extract a few bits from a large hex number, it is unwise to use this as Basic takes the top bit set to one as indicating a negative number, giving unexpected results. Use the logical operator AND to mask out the top bit first, if this is not required. E.g.:

```
&7F123456 DIV &10000
```

gives &7F12, but

```
&FF123456 DIV &10000
```

gives &FFFFFF13

Mike Kay

Pretty Easy List

Programs can be listed in a format similar to that produced by the 'Pretty List' program (BEEBUG Vol.3 No.9) using Wordwise. Spool the program onto disc or tape. Load it into Wordwise. Use menu option 5 to replace every colon with a pound sign and place an embedded 'DP13' at the start. Option 7 (or 6) will now produce a listing of the program with every statement on a new line. Unfortunately this will not work for programs with pound signs, or colons other than as statement separators.

Michael Keal

Half Procedures

If you want to use a section of an existing procedure as a procedure in its own right, without repeating the program lines, you can simply add a second DEF PROC statement. This does make the program somewhat unstructured but it is economic on memory. For example these two program sections are equivalent:

```
100 DEF PROCone
110 PRINT "one"
120 PRINT "two"
130 ENDPROC
140 :
150 DEF PROCtwo
160 PRINT "two"
170 ENDPROC
```

```
100 DEF PROCone
110 PRINT "one"
120 DEF PROCtwo
130 PRINT "two"
140 ENDPROC
```

N. Silver

Wordwise Plus Sentence Jumper

A simple sentence jumper can be programmed into a segment as follows:

```
FKEY 4,"."
CURSOR RIGHT 2
DISPLAY
```

Jonathan Temple

Decoding USR

The most difficult aspect of using the USR function is separating the four parts of the number returned into the accumulator, X register, Y register, and the status register. This is most easily done like this:

```
!&70=USR(address)
```

The accumulator will then be stored in location &70, Xin &71, Y in &72, and P in &73.

Jonathan Temple

High Precision Analogue

The analogue to digital converter in the Beeb uses three forward-biased diodes as its voltage reference source. These can drift severely with variations in temperature. For a more stable reference use a LM336Z reference zener diode connected with its '+' lead to Vref, its '-' lead to ground and its 'adjust' lead unconnected. This gives a reference voltage of about 2.49 volts with a maximum drift of 1.5mV over a 60 degree range.

Brian Edwards

Tube or Not

To determine, from within a program, if a 6502 second processor is connected to your Beeb, the following short routine may be used:

```
100 A%=&EA:Y%=&FF
110 IF USR(&FFF4) AND &
FF00 THEN tube%=TRUE ELSE t
ube%=FALSE
```

Richard Sterry

Light Fantastic

Colour Space does for colour what a music synthesizer does for sound — transforms the everyday into something magical. Geoff Bains has been controlling the lights with dazzling success.

Product : Colour Space
Supplier : Llamasoft
49, Mount Pleasant,
Tadley, Hants.
(07356) 4478
Price : £7.95

Colourspace, the packaging proclaims, is a light synthesiser. Colourspace is not a game. It is not a utility. It most certainly isn't a language. I haven't really decided just how it can be categorized. Just file under 'F for fun'.

Colourspace has to be experienced. The program starts in demo mode. The whole screen is filled with a mind-boggling dance of light and colour. Various sequences are played out on the screen until you retreat, with eyes still numb, to the instruction sheet to see what to do next.

Colourspace is all about playing with light and colour. A cursor is moved around under the control of the keyboard or joystick, and 'fire' will produce a pattern on the screen. This pattern will change with time in a kind of explosion of colour. Moving the cursor and pressing fire continuously causes a trail of blossoming light. To add to the effect there is a symmetrical repetition of your shape also moving around the screen.

This is impressive enough, but it's just the start. There are several 'presets' available, selected with the functions keys, each with its own pattern and associated effects. However, you can design your own effects. This is not easy, as the parameters to change are very alien to any normal activity.

The pattern can be in explosion mode, which gives extra umph to its dynamic quality, or in stroboscopic mode, which flashes it on and off - at a variable rate, naturally. The dynamic qualities of the pattern can be changed too. The various complicated controls allow you to alter the length of the trails produced by moving patterns, the colours that it changes through, not to mention the cursor speed and smoothness of the plotting. The symmetry can be changed between X, Y, or XY axis or quad reflection.

You can define your own patterns as well. Each stage of the changing pattern is drawn on the screen, pixel by pixel, with the joystick. You can even draw static graphics in the foreground of the screen and a very impressive pointed-eared gentleman's face is provided. The patterns and graphics can then be saved to disc or cassette for use at a later date. In fact sequences of movement can also be recorded in memory for effortless playback and saved to tape or disc too.

The instructions are as mind-blowing as the program. They are written by Jeff Minter, the flower power guru of Atari and Commodore software, and author of the original Colourspace. Although this version is not programmed by him, Minter's fondness for the hippy lifestyle makes the instructions strange reading. They are full of words like 'zowie', 'zarjaz', and 'froody' to express his obvious enthusiasm with this version.

This static textual review cannot do justice to this excellent package and it makes the program sound very clinical and technical, and so it can be if you're that way inclined. However, Colourspace is really just about kinetic art, impressing your friends with the power of computers, and having a great deal of fun.

Have some fun with Jonathan Temple's Talking Heads. Let your imagination run riot as you create sparkling conversations between your favourite characters. What did Uncle Clive really say to Alan Sugar?

Talking Heads is not a game in the strict sense of the word, but it can certainly provide quite a lot of amusement and entertainment. It also has its more serious uses, particularly in education, for creating imaginative conversations.

The program allows you to create two characters and then generate a conversation between them. However, other characters, once created, can be saved to cassette or disc, allowing you to build up a whole group of people who can then be made to talk to each other.

Now you can create the conversation you have only dreamed about. Why not re-create your friends, teachers, the rich or famous and then give them all those catch-phrases you know so well. Now as I was saying to Maggie the other day....

The program should be fairly straightforward to enter, and will work correctly with disc systems without resetting PAGE.

Whilst debugging the program, try to use option <9> to exit from the program as doing this will reset the cursor keys and Escape key to their default settings. You may also wish to leave out lines 150 and 240 (error trapping) until you have otherwise debugged the program.

CREATING CHARACTERS

Once the program has been entered the fun really begins. First of all, you will need to give your characters names, and to do this choose option <1> from the menu. You will be asked "Are you sure (Y/N)?" on doing this, and in fact all options that could destroy the data in the memory require a 'Y' answer before you can proceed further.

Once you have given the characters names (up to ten letters in length) you will be asked to enter starters for the two characters. Each may have up to four conversation starters, and these are used as opening gambits to start the ball rolling in a conversation, or to fall back on later.

When entering any text into the program, don't worry that the words are 'broken' on screen, as this will not be

seen when the conversation is displayed later. Pressing Return when prompted for the next starter will allow you to move on to the next character or back to the menu.

The next part of the character-building process involves giving each character a list of triggers and replies. If one of a character's triggers crops up in the conversation, it will 'trigger' off one of up to four stock replies. These triggers and replies should be entered using option <2> from the menu.

Whilst entering all this text you will probably make some mistakes - in this case use option <3> to edit them out. You will be asked which character to edit, and whether you want to edit the starters, or the triggers and replies.

When editing, use the cursor keys to move up and down and the Delete key to delete an entry. To insert a new entry (or alter an existing one) press Copy and then type in the new text. Note also, that when editing the triggers and replies, the left/right cursor keys allow movement from one trigger or reply to the next.

When entering the starters, triggers and replies, bear in mind that the starters and replies can be no more than sixty characters each in length and the triggers twenty.

THE OTHER MENU OPTIONS

Once you have entered and edited this data you can now start generating conversations; use option <4> to do this. Once a conversation has been generated it can be previewed with option <5> or printed out with option <6>. Previewing is performed in paged mode, so use Shift to scroll the next part of the conversation. Many different conversations can be generated from the same sentences, so it is well worth experimenting here.

If at some point you wish to save a character, or load in a previously saved one, then this can be done using options <7> and <8> respectively. The prefix 'C.' is automatically added to the filename requested so that all characters can be held in directory C for disc users. Option <9> allows you to leave the program.

Something else to note is that '*' commands (e.g. *CAT) can be entered from the menu - just press the * character and type the command in. However, disc users should be careful not to use commands which corrupt memory, such as *COMPACT or *COPY.

The Escape key will allow you to return to the menu from certain points - for instance the editing option or when prompted for the filename during load/save operations.

USING THE PROGRAM

Before seriously entering proper characters it is wise to experiment and become used to the program, although it is quite user-friendly and there are prompts throughout.

When creating characters, the starters, triggers and replies which you enter need to be carefully thought out to produce the best conversations. In fact, the quality of conversations produced by the program from the same characters can vary widely, from repetitive statements to interesting arguments. Therefore, it is always worth previewing and perhaps re-generating a conversation before printing it out, to get the best possible results.

The number of triggers and lines of conversation allowed are set by the variables MX and NL respectively in line 110 of the program. At present, MX is set to 20 for disc users, but tape users could increase this figure to 28 or perhaps 29 triggers. Those with a Second processor, B+ or Master 128 could set this figure even higher.

SPOKEN CONVERSATIONS

The program will also 'speak' the conversations using Superior Software's popular "Speech!" utility (see review this issue). To do this *RUN SPEECH before loading the Talking Heads program, or from the program's menu after loading, and the program will do the rest - use option <5> to hear the conversations spoken. This will not work on the Master or across the Tube - change line 140 to read 'speech=0'.

Because of memory limitations when the speech system is used, MX in line 110 will have to be set to around 11 or 12 for tape users, and disc users will have to set it even lower and possibly set PAGE to &1300. When using this facility it is a good idea

to set NL to a low number as well (perhaps 5 or so) because the speech can tend to be on the slow side!

MAGAZINE CASSETTE/DISC

This month's magazine cassette/disc contains two characters called Clive and Hermann for you to 'listen' to and experiment with.

```
   10 REM PROGRAM TALKING HEADS
   20 REM VERSION B0.1
   30 REM AUTHOR  J.Temple
   40 REM BEEBUG  June 1986
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 MODE 7
  110 MX=20:NL=20
  120 PROCinit:PROCclear
  130 REPEAT
  140 speech=(?&209<>&DF)
  150 ON ERROR PROCerror:GOTO 130
  160 C=FNmenu
  170 ON ERROR GOTO 3650
  180 IF C=1 PROCcreate
  190 IF C=2 PROCenter
  200 IF C=3 PROCedit
  210 IF C=4 PROCconversation
  220 IF C=5 PROCpreview(0,"Preview")
  230 IF C=6 PROCpreview(2,"Print")
  240 ON ERROR PROCerror:GOTO 130
  250 IF C=7 PROCfile(0,"Save")
  260 IF C=8 PROCfile(1,"Load")
  270 ON ERROR GOTO 3650
  280 UNTIL C=9
  290 *FX 4,0
  300 *FX 229,0
  310 END
  320 :
 1000 DEFPROCcreate
 1010 PROCclear
 1020 PROCtext(3,"Create new characters"
,2)
 1030 PRINT'G$;"Please enter first perso
n's name:-"'
 1040 N$(0)=FNinput(1,10,32,126)
 1050 PRINT'G$;"Please enter second pers
on's name:-"'
 1060 N$(1)=FNinput(1,10,32,126)
 1070 FOR P=0 TO 1:CLS
 1080 PROCtext(3,"Enter STARTERS for"+C$
+N$(P),2)
 1090 S=0:REPEAT
 1100 PRINTTAB(0,5+S*4);G$;"Please enter
 STARTER no.";S+1
 1110 B=0:REPEAT VDU B,31,0,6+S*4
 1120 S$=FNinput(0,60,32,126):B=7
 1130 IF S$="" IF S=0 E=TRUE
 1140 UNTIL E=0:S$(S,P)=S$:S=S+1
 1150 UNTIL S=4 OR S$="":NEXT:CC=TRUE
 1160 ENDPROC
 1170 :
 1180 DEFPROCenter
 1190 PROCtext(3,"Enter TRIGGERS and REP
LIES",2)
 1200 P=FNperson("Enter")
 1210 REPEAT
 1220 VDU 28,0,24,39,4,12,26
 1230 PRINTTAB(0,5);G$;"Please enter TRI
GGER no.";T%(P)
 1240 T$=FNinput(2,20,32,126)
 1250 IF E GOTO 1350
 1260 T$(T%(P),P)=T$:R=0:REPEAT
 1270 PRINTTAB(0,8+R*4);G$;"Please enter
 REPLY no.";R+1
 1280 B=0:REPEAT VDU B,31,0,9+R*4
 1290 R$=FNinput(0,60,32,126):B=7
 1300 IF R=0 AND R$="" E=TRUE
 1310 UNTIL E=0
 1320 R$(T%(P),R,P)=R$:R=R+1
 1330 UNTIL R=4 OR R$=""
 1340 T%(P)=T%(P)+1
 1350 UNTILT%(P)=MX+1 OR E=TRUE:TE=TRUE
 1360 ENDPROC
 1370 :
 1380 DEFPROCedit
 1390 PROCtext(3,"Edit character",2)
 1400 PRINT'G$;"Edit (S) STARTERS (T) TR
IGGERS ?";
 1410 B=0:REPEAT VDU B:G=GET AND &DF
 1420 B=7:UNTIL G=83 OR G=84
 1430 VDU G,10,10:S=(G=83):T=1
 1440 Y=5+ABS(S*4):H=Y
 1450 P=FNperson("Edit"):REPEAT
 1460 VDU 28,0,24,39,4,12,26
 1470 PRINT TAB(0,24);G$;"COPY to enter
 DELETE to delete";
 1480 IF S=0 PRINT TAB(0,4);G$;"TRIGGER
";T'" ";T$(T,P)
 1490 FOR J%=0 TO 3:VDU 26,31,0,8+J%*4
 1500 M$="REPLY":IF S M$="STARTER"
 1510 PRINT G$;M$;" ";J%+1;":"
 1520 VDU 28,1,12+J%*4,30,9+J%*4
 1530 IF S PRINT S$(J%,P) ELSE PRINT R$(
T,J%,P)
 1540 NEXT
 1550 VDU 26,31,1,H:REPEAT G=GET
 1560 IF G=139 VDU 31,1,VPOS-4:IF H=Y TH
EN VDU 31,1,21
 1570 IF G=138 VDU 31,1,VPOS+4:IF H=21 T
HEN VDU 31,1,Y
 1580 H=VPOS
 1590 UNTIL G=27 OR G=127 OR G=135 OR G=
136 OR G=137
 1600 IF G=127 IF FNce>1 PROCalter("")
 1610 IF G=135 PROCnew
 1620 IF G=136 T=T-1:IF T<1 T=T%(P)-1:IF
 T=0 T=1
 1630 IF G=137 T=T+1:IF T>=T%(P) T=1
```

```
 1640 UNTIL G=27
 1650 ENDPROC
 1660 :
 1670 DEFFNce
 1680 K%=0:FOR J%=0 TO 3
 1690 IF (S=-1 AND S$(J%,P)>"") OR (S=0
AND R$(T,J%,P)>"") THEN K%=K%+1
 1700 NEXT
 1710 =K%
 1720 :
 1730 DEFPROCnew
 1740 Z=VPOS:VDU 26,31,0,24,7
 1750 PRINTTAB(10);G$;"Type in new entry
";SPC(6);
 1760 VDU 31,1,Z
 1770 L$=FNinput(1,20-(VPOS>4)*40,32,126
)
 1780 IF NOT E PROCalter(L$)
 1790 ENDPROC
 1800 :
 1810 DEFPROCalter(M$)
 1820 IF E=TRUE ENDPROC
 1830 IF S=TRUE S$((VPOS-9)/4,P)=M$:ENDP
ROC
 1840 IF VPOS<9 T$(T,P)=M$:ENDPROC
 1850 R$(T,(VPOS-9)/4,P)=M$
 1860 ENDPROC
 1870 :
 1880 DEFPROCconversation
 1890 P=RND(2)-1:Q=1
 1900 S=-1:T=0:IFFNce<1 VDU7:ENDPROC
 1910 REPEAT:REPEAT J%=RND(4)-1
 1920 UNTIL S$(J%,P)>"":B$=S$(J%,P)
 1930 REPEAT
 1940 L$(Q)=N$(P)+STRING$(10-LEN(N$(P)),
" ")+": "+B$
 1950 VDU 30:P=P EOR 1
 1960 PROCtext(3,"GENERATING: "+STR$(Q),
2)
 1970 Q=Q+1:F=FNtrigger(B$)
 1980 IF F THEN B$=FNreply(F)
 1990 UNTIL F=FALSE OR Q=NL+1
 2000 UNTIL Q=NL+1:CG=TRUE
 2010 ENDPROC
 2020 :
 2030 DEFFNtrigger(L$)
 2040 K%=0:N%=0:REPEAT K%=K%+1
 2050 UNTIL INSTR(L$,T$(K%,P))>0 OR K%=T
%(P)-1
 2060 IF INSTR(L$,T$(K%,P))>0 N%=K%
 2070 =N%
 2080 :
 2090 DEFFNreply(N%)
 2100 REPEAT J%=RND(4)-1
 2110 UNTIL R$(N%,J%,P)>""
 2120 =R$(N%,J%,P)
 2130 :
 2140 DEFPROCpreview(N%,M$)
 2150 PROCtext(3,M$+" conversation",2)
 2160 VDU 28,0,24,39,4
 2170 IF N%=0 AND speech=0 VDU14
 2180 VDU N%
 2190 IF N%=2 PRINT"A conversation betwe
en "+N$(0)+" and "+N$(1)
 2200 FOR J%=1 TO Q-1
 2210 PRINT
 2220 PRINT L$(J%):IF speech PROCsay
 2230 NEXT:VDU 3,26,15:PRINT''
 2240 PROCtext(6,"   Press"+Y$+"SPACE BA
R"+C$+"to continue   ",0)
 2250 REPEAT UNTIL GET=32
 2260 ENDPROC
 2270 :
 2280 DEFPROCsay
 2290 $&700="*SAY "+MID$(L$(J%),12)
 2300 X%=0:Y%=&7:CALL &FFF7
 2310 ENDPROC
 2320 :
 2330 DEFPROCfile(N%,M$)
 2340 PROCtext(3,M$+" character data",2)
 2350 P=FNperson(M$)
 2360 PRINT'G$;"Enter filename for chara
cter data:-"'
 2370 F$="C."+FNinput(1,7,65,122)
 2380 IF E THEN ENDPROC
 2390 IF N%=0 F=OPENOUT(F$) ELSE F=OPENI
N(F$)
 2400 IF N%=0 PRINT #F,N$(P) ELSE INPUT
#F,N$(P)
 2410 FOR J%=0 TO 3
 2420 IF N%=0 PRINT #F,S$(J%,P) ELSE INP
UT #F,S$(J%,P)
 2430 NEXT
 2440 IF N%=0 PRINT #F,T%(P) ELSE INPUT
#F,T%(P)
 2450 FOR J%=1 TO T%(P)-1
 2460 IF N%=0 PRINT #F,T$(J%,P) ELSE INP
UT #F,T$(J%,P)
 2470 FOR K%=0 TO 3
 2480 IF N%=0 PRINT #F,R$(J%,K%,P) ELSE
INPUT #F,R$(J%,K%,P)
 2490 NEXT,
 2500 CLOSE #0:IFN% CC=TRUE:TE=TRUE
 2510 ENDPROC
 2520 :
 2530 DEFPROCerror
 2540 CLOSE #0:PRINT':REPORT
 2550 PRINT''C$;"Press any key";:G=GET
 2560 ENDPROC
 2570 :
 2580 DEFFNperson(M$)
 2590 PRINT'G$;M$;" data for character (
1/2) ? ";
 2600 B=0:REPEAT VDU B:G=GET AND &CF
 2610 B=7:UNTIL G=1 OR G=2:CLS
 2620 IF M$="Save" M$="Sav"
 2630 N$=M$+"ing data"
 2640 IF M$<>"Load" N$=N$+" for "+C$+N$(
G-1)
 2650 PROCtext(3,N$,2)
```

```
2660 =G-1
2670 :
2680 DEFFNinput(V%,W%,X%,Y)
2690 Z=VPOS:VDU 28,1,Z+2,39,Z
2700 IF W%=60 VDU28,1,Z+2,30,Z,12
2710 L$="":REPEAT L%=LEN(L$)
2720 A=GET
2730 IF A=27 E=TRUE:A=13:IF L%=0 GOTO 2
800
2740 E=FALSE
2750 IF A=127 IF L%>0 L$=LEFT$(L$,L%-1)
:GOTO 2790
2760 IF A=13 IF L%>=V% GOTO 2790
2770 IF A<32 OR L%=W% A=7
2780 IF A>=X% AND A<=Y L$=L$+CHR$(A) EL
SE A=7
2790 VDU A
2800 UNTIL A=13
2810 VDU 26:IF L$>"" VDU 31,0,Z+2
2820 =L$
2830 :
2840 DEFFNsure
2850 PRINT TAB(0,22);
2860 PROCtext(1,"Are you sure (Y/N) ?",
0)
2870 B=0:REPEAT VDU B:Z=GET AND &DF
2880 B=7:UNTIL Z=78 OR Z=89
2890 =(Z=89)
2900 :
2910 DEFFNmenu
2920 REPEAT:REPEAT CLS
2930 PROCtext(3,"TALKING HEADS",2)
2940 VDU 11
2950 PROCtext(6,"by Jonathan Temple 198
6",1)
2960 RESTORE 3550
2970 FOR J%=1 TO 9:READ M$
2980 PRINTTAB(0,2*J%+3);G$;J%;")  ";M$
2990 NEXT
3000 PRINTTAB(0,23);C$;"Enter option (1
-9) ?";
3010 B=0:REPEAT VDU B:G=GET AND &CF
3020 B=7:UNTIL G>0 AND G<11
3030 IF G=10 PROCoscli
3040 UNTIL G>0 AND G<10
3050 VDU31,0,G*2+3,136,31,0,23
3060 F=TRUE:IF G=1 OR G>7 F=FNsure
3070 IFG>1 IFG<8 IFCC=0 PROCerr("No cha
racters created"):F=0:GOTO3100
3080 IF G=4 IF TE=0 PROCerr("No trigger
s entered"):F=0
3090 IF G=5 OR G=6 IF CG=0 PROCerr("No
conversation generated"):F=0
3100 UNTIL F:CLS
3110 =G
3120 :
3130 DEFPROCerr(E$)
3140 PRINT''E$''C$;"Press any key";
3150 Z=GET
3160 ENDPROC
3170 :
3180 DEFPROCoscli
3190 PRINTTAB(1,23);"*";SPC(20);TAB(2,2
3);
3200 $&700=FNinput(1,39,32,126)
3210 IF NOT FNsure ENDPROC
3220 CLS:X%=0:Y%=&7:CALL &FFF7
3230 PRINT''C$;"Press any key";:G=GET
3240 ENDPROC
3250 :
3260 DEFPROCtext(C,T$,D%)
3270 PRINT:FOR L%=1 TO D%:VDU 128+C
3280 PRINTTAB((34-LEN(T$))/2);CHR$(139+
D%);T$
3290 NEXT
3300 ENDPROC
3310 :
3320 DEFPROCclear
3330 FOR J%=0 TO 1:T%(J%)=1
3340 N$(J%)=STRING$(10," ")
3350 FOR K%=0 TO 3
3360 S$(K%,J%)=STRING$(60," ")
3370 S$(K%,J%)="":NEXT
3380 FOR K%=1 TO MX
3390 T$(K%,J%)=STRING$(20," ")
3400 FOR L%=0 TO 3
3410 R$(K%,L%,J%)=STRING$(60," ")
3420 R$(K%,L%,J%)="":NEXT,,
3430 FOR K%=1 TO NL
3440 L$(K%)=STRING$(76," "):NEXT
3450 CC=FALSE:TE=FALSE:CG=FALSE
3460 ENDPROC
3470 :
3480 DEFPROCinit
3490 DIM L$(NL),N$(1),S$(3,1),T$(MX,1),
R$(MX,3,1),T%(1)
3500 G$=CHR$130:Y$=CHR$131:C$=CHR$134
3510 *FX 4,1
3520 *FX 229,1
3530 ENDPROC
3540 :
3550 DATA Create new characters
3560 DATA Enter more data
3570 DATA Edit present data
3580 DATA Generate new conversation
3590 DATA Preview conversation
3600 DATA Print out conversation
3610 DATA Save character data
3620 DATA Load character data
3630 DATA Leave program
3640 :
3650 MODE 7:CLS:REPORT
3660 PRINT " at line ";ERL:*FX 4,0
3670 *FX 229,0
```

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.

Editor: Mike Williams
Production Assistant: Yolanda Turuelo
Technical Assistant: Alan Webster
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to Sheridan Williams, Adrian Calcraft, Geoff Bains, John Yale and Tim Powys-Lybbe.

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility, whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Publications Limited.

BEEBUG Publications Ltd (c) 1986

Editorial Address

BEEBUG
PO BOX 50,
Holywell Hill,
St. Albans AL1 3YS

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet, 'Notes of Guidance for Contributors' is available on receipt of an A5 (or larger) SAE.

In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", or other means, but please ensure an adequate written description of your contribution is also included. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

Please send all editorial material to the editorial address above. If you require a reply it is essential to quote your membership number and enclose an SAE.

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, subscription queries and orders for back issues to the subscriptions address.

MEMBERSHIP SUBSCRIPTION RATES

£ 6.90 6 months (5 issues) UK ONLY)
£12.90 UK - 1 year (10 issues)
£19.00 Europe, £22.00 Middle East
£24.00 Americas & Africa, £26.00 Elsewhere

BACK ISSUES
(Members only)

Vol	Single issues	Volume sets (10 issues)
1	90p	£8
2	£1	£9
3	£1.20	£11
4	£1.20	–
5	£1.30	–

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK	30p	10p
Europe	70p	20p
Elsewhere	£1.50	50p

All overseas items are sent airmail (please send a sterling cheque). We will accept official UK orders but please note that there will be a £1 handling charge for orders under £10 that require an invoice. Note that there is no VAT on magazines.

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

Subscriptions, Back Issues & Software Address

BEEBUG
PO BOX 109
St. Johns Road
High Wycombe HP10 8NP

Hotline for queries and software orders

St. Albans (0727) 40303
Manned Mon-Fri 9am-4.30pm

24hr Answerphone Service for Access and Barclaycard orders, and subscriptions
Penn (049481) 6666

If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

Magazine Cassette/Disc

JUNE 1986 CASSETTE/DISC CONTENTS

COMPUTER SIMULATION — random sampling routine and complete application.

BEEBUG FILER GOES GRAPHIC — full program to display all your data graphically.

THE MASTER SERIES — open up the private RAM with these routines to save and load function key definitions.

PASSING ARRAYS TO PROCEDURES — highly useful techniques for Basic programmers.

BEEBUG WORKSHOP — examples of how to speed up programs in real style.

FIRST COURSE — useful routines for using and testing the ADVAL function.

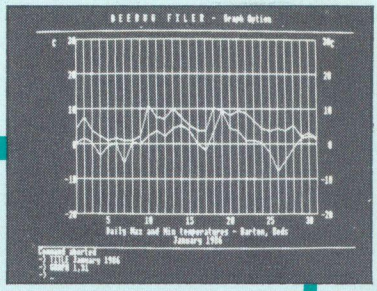
HARDWARE ANIMATION — complete demonstration of this novel graphics technique

TALKING HEADS — have fun by creating characters and conversations between the rich and the famous, friends and relatives, pop-stars and politicians.

EXTRA FEATURES THIS MONTH

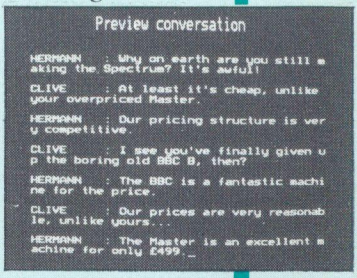
MAGSCAN — data for this issue of BEEBUG (Vol. 5. No. 2)

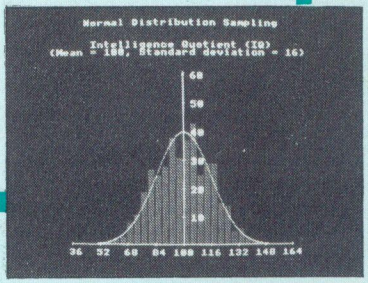
GRID RUNNER 2 — highly playable follow-up to Grid Runner (on the BEEBUG demo disc). A superb example of machine code graphics.



Filer Graphics

Talking Heads





Computer Simulation

All this for £3.00 (cass) £4.75 (disc) +50p p&p.
Back issues (disc since Vol. 3 No. 1, cass since Vol. 1 No. 10) available at the same prices.

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25.50	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to disc subscription on receipt of £1.70 per issue of the subscription left to run.

All subscription and individual orders to
BEEBUG, PO BOX 109, St. Johns Road, High Wycombe HP10 8NP

Why Pay £2.50 for a FREE Demo Disc?

This month's code-breaker number is X95130

This offer is limited to one disc per household/institution

Please send me a "free" disc,
I enclose a cheque for £2.50/Please debit my Barclaycard/Access No. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
OR I also enclose an order for software, please charge only £1.50

Name

Address

Specify 40 or 80 Track

Send to:
Beebugsoft,
PO Box 109,
High Wycombe,
Bucks., HP10 8NP